3 Phát triển hệ thống

Mục đích

Phát triển hệ thống là việc tạo nên phần mềm để vận hành máy tính. Thông thường, công việc này được thực hiện lần lượt từ phân tích yêu cầu, thiết kết ngoài, thiết kế trong đến lập trình và kiểm thử. Tuy vậy, cũng có nhiều phương pháp luận đã được đề xuất tùy thuộc vào từng ngữ cảnh phát triển hệ thống. Trong phần 1, chúng ta sẽ học về các phương pháp luận cho việc phát triển hệ thống cùng các yếu tố hỗ trợ như: các ngôn ngữ lập trình, các nhóm công cụ và đánh giá chất lượng phần mềm. Trong phần 2, chúng ta sẽ học về những thủ tục cụ thể của việc phát triển hệ thống và các phương pháp kiểm thử.

3.1 Các phương pháp phát triển hệ thống
3.2 Các công việc trong các qui trình phát triển hệ thống

[Các thuật ngữ và khái niệm cần nắm vững]
Ngôn ngữ lập trình, trình biên dịch, chương trình con, đệ qui, đồng sử dụng, CASE, ERP, mô hình thác đổ, mô hình chế thử, phương pháp điểm chức năng, DFD, biểu đồ E-R, kiểm điểm, kiểm thử hộp trắng, kiểm thử hộp đen, sự độc lập của các mô-đun

3.1 Các phương pháp phát triển hệ thống

Mở đầu

Để phát triển hệ thống, chúng ta cần biết đến các phương pháp luận của việc phát triển hệ thống. Những phương pháp luận này có thể được phân loại thành các mô hình qui trình và các mô hình chi phí. Mô hình qui trình là phương pháp của các thủ tục phát triển trong khi mô hình chi phí là phương pháp của việc đánh giá chi phí. Để áp dụng những phương pháp này, trước hết chúng ta cần biết về môi trường phát triển hệ thống. Một môi trường phát triển hệ thống là một nhóm các công cụ hỗ trợ cho việc phát triển hệ thống, bao gồm các ngôn ngữ lập trình và CASE.

3.1.1 Các ngôn ngữ lập trình

Điểm chính	➢ Các loại ngôn ngữ lập trình bao gồm: ngôn ngữ thủ tục, ngôn ngữ chức năng, ngôn ngữ logic và hướng đối tượng. ➢ Các ngôn ngữ tiêu biểu bao gồm COBOL, C, Java và SGML.

Ngôn ngữ lập trình là ngôn ngữ mô tả các tiến trình (chương trình) mà ta muốn máy tính thực hiện. Chúng ta chọn các ngôn ngữ lập trình thích hợp tùy thuộc vào từng ứng dụng.

Phân loại các ngôn ngữ lập trình

Dưới đây là một cách phân loại các ngôn ngữ lập trình và một số ngôn ngữ tiêu biểu.

Loại	Đặc điểm	Các ngôn ngữ lập trình
Thủ tục (Procedural[1])	Các thủ tục được biểu diễn dưới dạng các thuật toán cụ thể. Mỗi thủ tục được viết ra sẽ được thực thi bởi máy tính, một lệnh tại một thời điểm.	COBOL, C, Fortran, Pascal,...
Chức năng (Functional)	Các bước tiến trình được biểu diễn bằng cách kết hợp của các chức năng cơ bản (xử lí danh sách - list processing)	Lisp,...
Logic	Các mối quan hệ được định nghĩa bởi các	Prolog,...

[1] **Phi thủ tục (Non-procedural)**: Có một số ngôn ngữ lập trình không theo hướng thủ tục và được gọi là các ngôn ngữ lập trình phi thủ tục. Chúng được nhận biết bởi đặc trưng là thứ tự các lệnh được viết ra trong chương trình không đúng với thứ tự thực thi. Thông thường các tham số sẽ được đưa ra và các tiến trình sẽ được thực thi theo nội dung của các định nghĩa tham số.

(Logic)	hàm logic cơ bản (xử lí suy diễn - inferential processing)	
Hướng đối tượng (Object-oriented)	Thao tác được kiểm soát bởi các đối tượng. Chúng kết hợp dữ liệu với việc xử lí.	Java, C++, Smalltalk,…

Các ngôn ngữ lập trình

Đặc điểm của các ngôn ngữ lập trình thông dụng được mô tả như dưới đây.

Hướng thủ tục/chức năng/logic/đối tượng

Ngôn ngữ	Đặc điểm
COBOL	Ngôn ngữ xử lí thương mại (business-processing language) Đặc tả về ngôn ngữ này được tạo nên bởi CODASYL.
C	Được phát triển bởi AT&T[2] để viết nên hệ điều hành cho UNIX[3] Có tính khả chuyển (portability) cao
Fortran	Được phát triển bởi IBM như là một ngôn ngữ tính toán cho khoa học và công nghệ
Pascal	Ngôn ngữ lập trình cấu trúc được phát triển với mục đích giảng dạy cho sinh viên
Lisp	Ngôn ngữ xử lí danh sách được phát triển tại MIT[4] Được sử dụng cho việc nghiên cứu trong lĩnh vực trí tuệ nhân tạo,…
Prolog	Ngôn ngữ có cơ chế suy diễn Được phát triển tại Đại học Marseille của Pháp
C++	Ngôn ngữ hướng đối tượng và là sự mở rộng của C Hoàn toàn tương thích ở mức cao hơn với C
Java	Được phát triển bởi Sun Microsystems, dựa trên C++ Chạy trên bất kì hệ điều hành nào
Smalltalk	Được phát triển bởi Xerox tại phòng thí nghiệm Palo Alto Theo kiểu đối thoại và lập trình được

Các ngôn ngữ đánh dấu (các ngôn ngữ định dạng văn bản)

Đây là các ngôn ngữ mà ở đó thông tin về bố cục, cỡ phông, định dạng và các đặc tả khác được gắn trực tiếp vào đó để hiển thị lên màn hình hoặc phục vụ việc in ấn. Việc chèn thêm các biểu tượng (tag – thẻ) như <TITLE> và </TITLE> trong một đoạn được gọi là đánh dấu hay gắn thẻ (tagging). Bảng dưới đây minh họa các ngôn ngữ đánh dấu chính:

Ngôn ngữ	Đặc điểm
SGML	Standard Generalized Markup Language (Ngôn ngữ đánh dấu tổng quát hóa chuẩn) Cấu trúc logic và cấu trúc ngữ nghĩa của tài liệu được biểu diễn với các dấu đơn giản.
HTML	HyperText Markup Language (Ngôn ngữ đánh dấu siêu văn bản)

[2] **AT&T**: American Telephone and Telegraph, một công ty điện báo lâu đời nhất thế giới và lớn nhất tại Mĩ.
[3] (Chú ý) C là ngôn ngữ lập trình được phát triển để viết nên hệ điều hành cho UNIX, nhưng vì nó dễ sử dụng nên ngày nay có một lượng lớn các chương trình được viết bởi C, bao gồm các ứng dụng thương mại và các hệ điều hành.
[4] **MIT**: Massachusetts Institute of Technology.

	Đây là ngôn ngữ được sử dụng trong việc tạo các trang web trên Internet.
XML	eXtensible Markup Language (Ngôn ngữ đánh dấu có khả năng mở rộng) Đây là một sự mở rộng của chức năng siêu liên kết HTML, được mở rộng tới mức SGML có thể được gửi và nhận qua mạng. Các thẻ không cố định mà có thể được định nghĩa tự do.

Các ngôn ngữ lập trình khác

Bảng dưới đây mô tả về các ngôn ngữ lập trình khác

Ngôn ngữ	Đặc điểm
PostScript	Là một ngôn ngữ mô tả trang[5] được phát triển bởi Adobe Systems của Mĩ.
Visual Basic	Là một ngôn ngữ lập trình cho Windows, được phát triển bởi Microsoft của Mĩ.
Perl	Là một ngôn ngữ kịch bản mô tả việc truy cập các bộ đếm và CGI[6] của các trang web.

Các ngôn ngữ kịch bản (Script languages)

Ngôn ngữ kịch bản là ngôn ngữ sử dụng văn bản (kí tự) để mô tả các thủ tục được thực thi bởi máy tính. Các thủ tục xử lí được biểu diễn bởi một ngôn ngữ kịch bản được gọi là các **script**. Phần lớn chúng nằm trong các phần mềm cơ sở dữ liệu cũng như phần mềm bảng tính và được sử dụng như các macro. Theo cách những ngôn ngữ này mô tả các thủ tục thì chúng giống như các ngôn ngữ lập trình hướng thủ tục; tuy nhiên, các script này có đặc trưng là điều khiển theo sự kiện (event-driven)[7]. Ngoài ra, một môi trường phát triển sử dụng GUI thường được cung cấp để hỗ trợ cho người dùng cuối giúp họ viết các chương trình một cách dễ dàng.[8]

3.1.2 Cấu trúc chương trình và chương trình con

Điểm chính	➢ Các cấu trúc chương trình bao gồm các loại: đồng sử dụng (reentrance), tái sử dụng và đệ qui. ➢ Các chương trình con có thể là các chương trình con mở hoặc đóng.

Các tiến trình được sử dụng thường xuyên trong một chương trình hoặc các tiến trình chia sẻ dung chung cho nhiều chương trình được duy trì như

[5] **Ngôn ngữ mô tả trang**: Đây là ngôn ngữ được sử dụng để định nghĩa ảnh in ra cho máy in khi in văn bản sử dụng máy in trang. Các ảnh đó có thể được in ra giống nhau ngay cả khi các máy in có những độ phân giải khác nhau..

[6] **CGI (Common Gateway Interface)**: Đây là cơ chế lấy các yêu cầu từ một trình duyệt WWW, gọi một chương trình bên ngoài được yêu cầu và trả về các kết quả thực thi cho trình duyệt WWW.

[7] **Điều khiển theo sự kiện**: Là chương trình được kích hoạt bởi một sự kiện và khởi động để đáp ứng và xử lí sự kiện. Một sự kiện là bất kì một sự thay đổi có điều kiện nào, chẳng hạn như việc nhấn vào bàn phím. Các chương trình khởi động khi người dùng nhấp chuột vào một biểu tượng là chương trình điều khiển theo sự kiện.

[8] (FAQ) Có các câu hỏi để tổng hợp và phân loại các ngôn ngữ phổ biến. Chẳng hạn, ta biết rằng COBOL là ngôn ngữ thủ tục, Lisp là chức năng, và Java là hướng đối tượng.

những chương trình riêng rẽ và được chia sẻ cho nhiều chương trình. Những chương trình như vậy được gọi là chương trình con và các cấu trúc khác nhau được sử dụng tuỳ theo các điều kiện sử dụng.

Cấu trúc chương trình

Theo cấu trúc, các chương trình có thể được phân loại như dưới đây:

Đệ qui	Cấu trúc gọi lại chính nó
Tái sử dụng	Có thể được sử dụng lặp đi lặp lại mà không cần nạp lại
Đồng sử dụng	Nhiều tác vụ[9] có thể sử dụng chương trình ở cùng một thời điểm
Tái sử dụng tuần tự	Nhiều tác vụ có thể sử dụng chương trình một cách tuần tự
Không dùng lại được	Phải nạp lại mỗi khi sử dụng

Đệ qui

Một thủ tục được gọi là **đệ qui** nếu định nghĩa của nó tham chiếu tới chính nó. Chương trình định nghĩa của một chương trình con hay một hàm sử dụng chính chương trình con hoặc hàm đó được gọi là **chương trình đệ qui**. Mỗi tham chiếu như vậy được biết đến như một **lời gọi đệ qui.** Lời gọi này có thể được sử dụng trong hầu hết các ngôn ngữ lập trình ngoại trừ COBOL và Fortran.

Tái sử dụng (reusable)

Thuật ngữ này nói tới loại cấu trúc chương trình cho phép nhiều chương trình (hay tác vụ) chia sẻ việc sử dụng chương trình này mà không cần nạp lại vào bộ nhớ chính ở mỗi thời điểm. Nếu chương trình có thể được sử dụng đồng thời bởi nhiều tác vụ thì nó được gọi là **đồng sử dụng**[10] (reentrant); nếu không, nó được gọi là **tái sử dụng tuần tự** (serially reusable).

Chương trình con

Chương trình con là một phần của chương trình được sử dụng lặp đi lặp lại

[9] **Tác vụ**: Là một đơn vị xử lí có được khi các tiến trình được chia nhỏ rất chi tiết.

[10] (Chú ý) Trong một chương trình đồng sử dụng, các phần không thể thay đổi (chủ yếu là các phần thủ tục) và các phần có thể thay đổi (chủ yếu là dữ liệu) được phân tách sao cho các chương trình có thể sử dụng nó ở cùng thời điểm nhờ chia sẻ việc sử dụng những phần không thể thay đổi và giữ lại những phần có thể thay đổi theo chương trình gọi chương trình đồng sử dụng. Thông thường, hầu hết các chương trình xử lí trực tuyến đều là các chương trình có cấu trúc đồng sử dụng.

bên trong chương trình để thực thi các thủ tục chung. Nếu có nhiều chương trình cùng thực thi một số thủ tục thì những thủ tục này có thể gom lại như một chương trình và những chương trình khác có thể chia sẻ việc sử dụng nó. Những chương trình như vậy cũng được gọi là chương trình con.

Chương trình con mở

Một chương trình con mở[11] là một chương trình con được nhúng vào bất kì nơi nào chương trình cần với số lần tùy thích.

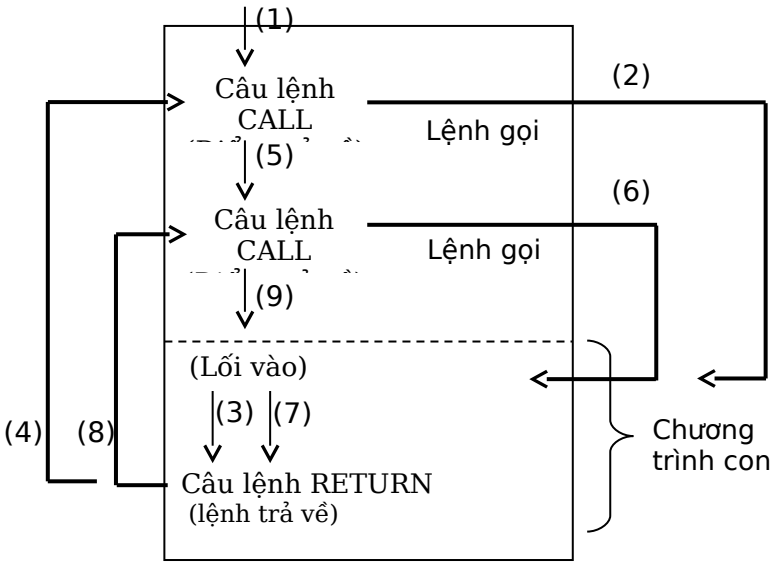
Chương trình con đóng

Một chương trình con đóng được tạo ra độc lập với các chương trình cần tới nó. Nếu một chương trình cần chương trình con này, nó thực hiện một lời gọi chương trình con (thường là câu lệnh CALL) để chuyển điều khiển tới chương trình con.

Hình dưới đây minh họa khái niệm về một chương trình con đóng. Các tiến trình được thực thi theo thứ tự (1), (2), (3),... Bằng câu lệnh CALL, chương trình nhảy tới lối vào chương trình con, và bằng câu lệnh RETURN, nó trả về vị trí của câu lệnh CALL (điểm trả về).[12]

[11] **Chương trình con mở**: Có thể được thực thi như một macro trong hợp ngữ, một thư viện sao chép trong COBOL, và "%include" trong C.

[12] (FAQ) Về cấu trúc chương trình, gần đây có nhiều câu hỏi liên quan đến đệ qui và đồng sử dụng. Đệ qui là gọi lại chính nó còn đồng sử dụng là được gọi đồng thời bởi nhiều chương trình.


(1)
Câu lệnh
CALL
(2)
Lệnh gọi
(5)
(6)
Câu lệnh
CALL
Lệnh gọi
(9)
(Lối vào)
(3)
(7)
(4)
(8)
Câu lệnh RETURN
(lệnh trả về)
Chương
trình con

3.1.3 Các bộ xử lí ngôn ngữ

Điểm chính	➢ Các bộ xử lí ngôn ngữ bao gồm các trình biên dịch, các trình thông dịch… ➢ Các mô-đun nạp được tạo ra bởi các bộ soạn thảo liên kết.

Một ngôn ngữ lập trình sử dụng cách biểu diễn tương tự như những ngôn ngữ thường ngày để có thể giúp cho việc viết các chương trình một cách dễ dàng. Tuy nhiên, máy tính không thể hiểu các câu lệnh của bất kì ngôn ngữ lập trình nào. Do đó, cần phải chuyển đổi những chương trình được viết bởi các ngôn ngữ lập trình thành định dạng mà máy tính có thể hiểu được. Việc chuyển đổi này được thực thi bởi bộ xử lí ngôn ngữ.[13]

Cá bộ xử lí ngôn ngữ

Bộ xử lí ngôn ngữ

bộ xử lí ngôn ngữ là một chương trình phiên dịch (chuyển đổi) các ơng trình nguồn thành ngôn ngữ máy. Các bộ xử lí ngôn ngữ được trình dưới đây:

Assembler	Trình dịch hợp ngữ: Dịch chương trình hợp ngữ thành ngôn ngữ máy.
Compiler	Trình biên dịch: Dịch ngôn ngữ biên dịch[14] thành ngôn ngữ máy (COBOL, Fortran, C,…)
Generator	Trình sinh mã: Tạo các chương trình bằng cách cho các tham số (RPG,…)
Interpreter	Trình thông dịch: Thực thi khi dịch các câu lệnh (BASIC, APL, LOGO,…)

Ngoài ra, cũng có những bộ tiền xử lí[15] chuyển đổi chương trình nguồn thành một ngôn ngữ biên dịch, không phải ngôn ngữ máy.

Các thủ tục của trình biên dịch

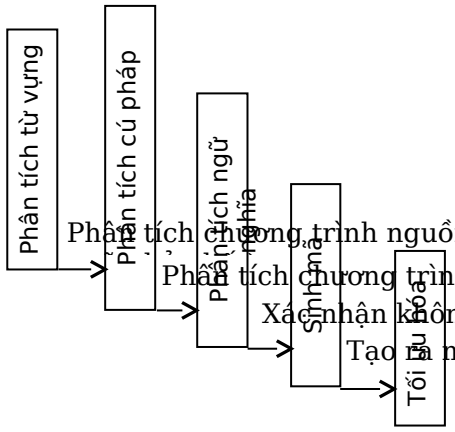
Một ngôn ngữ biên dịch được dịch thành ngôn ngữ máy theo thứ tự dưới đây. Chương trình đã được dịch thành ngôn ngữ máy được gọi là chương

[13] (FAQ) Có nhiều câu hỏi về việc lựa chọn đâu là đặc điểm của thông dịch và biên dịch. Hãy chắc chắn rằng bạn đã hiểu đầy đủ những đặc điểm đó. Các câu hỏi liên quan đến thủ tục của trình biên dịch cũng thường được hỏi trong các kì thi.

[14] **Ngôn ngữ biên dịch**: Là một ngôn ngữ lập trình tạo ra các chương trình đích từ chương trình nguồn nhờ sử dụng trình biên dịch. Nó cũng được gọi là một ngôn ngữ cấp cao và bao gồm COBOL, Fortran, Pascal, PL/I, và C. Ngôn ngữ biên dịch sử dụng cách biểu diễn giống như trong đời sống thường ngày của con người. Vì vậy chúng dễ hiểu và dễ học.

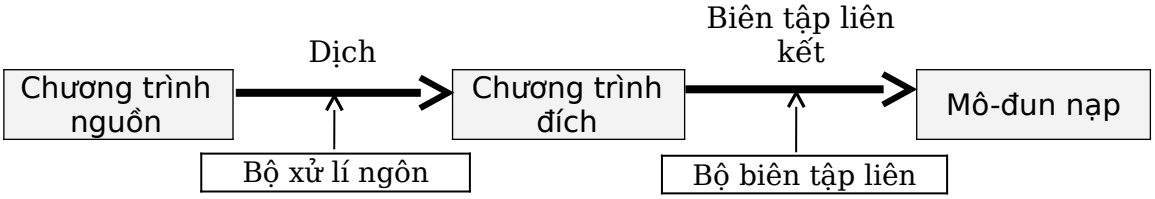
[15] **Bộ tiền xử lí**: Là chương trình lấy chương trình nguồn trước khi chúng được trình biên dịch dịch sang ngôn ngữ máy và bắt chúng thực thi với nhiều tiến trình. Chẳng hạn, một bộ tiền xử lí cho ngôn ngữ C hỗ trợ các chức năng như định nghĩa các giá trị số được tìm thấy trong chương trình nguồn dưới dạng các chuỗi kí tự và lấy các tệp thư viện được tham chiếu bởi chương trình nguồn. Chúng được đánh dấu bằng từ khóa “include”.

trình đích (hay mô-đun đích).


Phân tích từ vựng
Phân tích cú pháp
Phân tích ngữ nghĩa
Sinh mã
Tối ưu mã
Phân tích chương trình nguồn thành các biến và các token (các đơn vị ngôn
Phân tích chương trình theo cú pháp của ngôn ngữ
Xác nhận không có lỗi ngữ nghĩa trong chương trình
Tạo ra mã ở mức ngôn ngữ máy
Loại bỏ những phần không cần thiết ở chương trình đích
Phép nhân được thay thế bởi phép cộng liên tiếp
Những biến không cần thiết trong vòng lặp được đưa ra ngoài vòng lặp

Tạo mô-đun nạp

Máy tính chỉ có thể đọc và thực thi ngôn ngữ máy, do vậy bất kì chương trình được viết bằng ngôn ngữ nào khác cần phải được dịch thành ngôn ngữ máy. Một cách để thực hiện việc đó là sử dụng trình biên dịch.[16]



Các mô-đun nạp là các chương trình có thể thực thi. Các chương trình đích (được dịch bởi một bộ xử lí ngôn ngữ) không thể thực thi được. Qua một chương trình biên tập liên kết (bộ biên tập liên kết), những gì cần cho việc thực thi sẽ được bổ sung vào chương trình đích.

Trong việc liên kết hai hay nhiều chương trình đích, **bộ biên tậpliên kết** sẽ gọi các hàm và các chương trình con mà chương trình đích sử dụng từ thư viện phần mềm và liên kết chúng vào chương trình đích. Bộ biên tập liên kết cũng được gọi là **bộ liên kết**.

Các trình thông dịch không có chương trình đích. Nói đúng hơn, chúng thực thi các câu lệnh giống như việc dịch từng câu lệnh. Các trình sinh mã tạo trực tiếp các mô-đun nạp bằng cách cho các tham số.

Thực thi chương trình

Để thực thi một chương trình, cần phải lưu chương trình đó trong bộ nhớ chính hoặc trong một bộ nhớ ảo. Chức năng này được thực hiện bởi một bộ nạp.

Bộ nạp lưu một mô-đun nạp trong bộ nhớ chính và sau đó ở mỗi thời điểm, máy tính sẽ lấy một lệnh từ mô-đun nạp, dịch và thực thi nó.

[16] (Chú ý) Việc liên kết một chương trình con trong khi tạo mô-đun nạp được gọi là liên kết tĩnh. Ngược lại, nếu cần cũng có thể liên kết một chương trình con trong khi chương trình được thực thi. Phương pháp này được gọi là liên kết động.

3.1.4 Các môi trường phát triển và các gói phần mềm

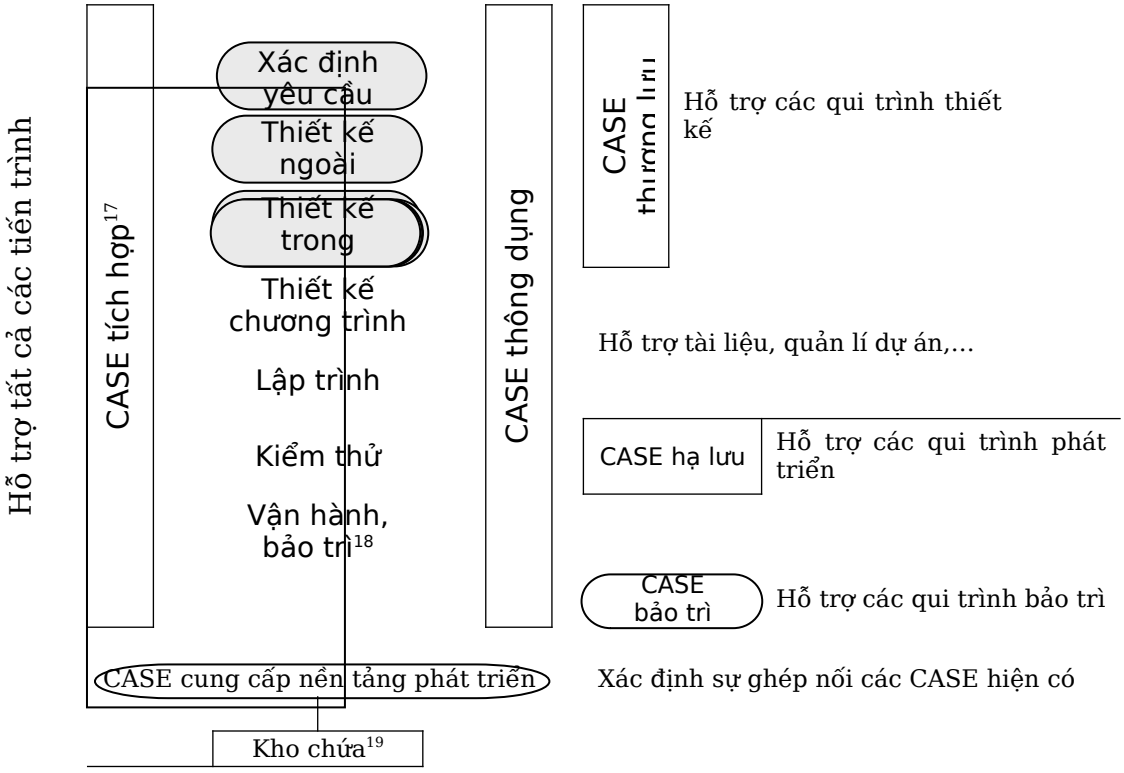
Điểm chính	➢ Các công cụ CASE và công cụ hỗ trợ kiểm thử sẽ hỗ trợ cho phát triển hệ thống. ➢ ERP là một gói phần mềm được thiết kế để làm cho các qui trình kinh doanh hiệu quả hơn.

Môi trường phát triển bao gồm các phần cứng cần thiết để xây dựng hệ thống và các phần mềm chẳng hạn như các công cụ hỗ trợ phát triển hệ thống.

CASE (Computer Aided Software Engineering)

CASE là một nhóm các phần mềm hỗ trợ việc phát triển hệ thống và tự động hóa các công việc bảo trì. CASE bao gồm các cơ sở dữ liệu dung chung - nơi lưu trữ những thông tin cần thiết cho việc phát triển chẳng hạn như các yêu cầu và thông tin thiết kế cho việc phát triển hệ thống. Nó cũng thực hiện việc quản lí một cách thống nhất toàn bộ qui trình phát triển hệ thống. Hơn nữa, các kết quả thiết kế có thể được minh họa bởi những hình vẽ dễ hiểu.

CASE cụ thể (Hỗ trợ các qui trình cụ thể)


Hỗ trợ tất cả các tiến trình
CASE tích hợp[17]
Xác định yêu cầu
Thiết kế ngoài
Thiết kế trong
Thiết kế chương trình
Lập trình
Kiểm thử
Vận hành, bảo trì[18]
CASE thông dụng
CASE thượng lưu
Hỗ trợ các qui trình thiết kế
Hỗ trợ tài liệu, quản lí dự án,...
CASE hạ lưu
Hỗ trợ các qui trình phát triển
CASE bảo trì
Hỗ trợ các qui trình bảo trì
CASE cung cấp nền tảng phát triển
Xác định sự ghép nối các CASE hiện có
Kho chứa[19]

[17] **CASE tích hợp:** Là những công cụ hỗ trợ toàn bộ qui trình phát triển hệ thống. Ban đầu, ý tưởng của CASE tích hợp là có một CASE bao phủ được tất cả các qui trình; tuy nhiên, thực tế là CASE từng phần đã được sử dụng và ý tưởng tốt hơn hết là sử dụng những công cụ có sẵn này đã trở nên phổ biến. Vì vậy, ngày nay CASE tích hợp được phát triển như một cách để tạo nên sự ghép nối giữa các công cụ CASE mà nhờ đó, các thông tin thiết kế có thể được trao đổi tốt hơn.

[18] (Gợi ý) Một số CASE thông dụng có những chức năng hỗ trợ toàn bộ qui trình phát triển. Tuy nhiên, chúng được phân biệt với CASE tích hợp. CASE thông dụng quản lí các lĩnh vực bên ngoài thông tin thiết kết, chẳng hạn như hỗ trợ tài liệu (bảng, đồ thị, hình vẽ), quản lí dự án, quản lí cấu hình hệ thống.

[19] **Kho chứa**: Là một cơ sở dữ liệu các công cụ CASE để lưu trữ nhiều loại thông tin, nó cũng được biết đến như một cơ sở dữ liệu hoặc mộ bộ lưu trữ công nghệ phần mềm hay một thiết bị lưu trữ. Với việc quản lí thông tin thiết kế một cách thống nhất nhờ sử dụng kho chứa, ta có thể kiểm tra tính nhất quán và toàn vẹn cũng như tự động hóa các qui trình phát triển.

Các công cụ hỗ trợ kiểm thử

Các công cụ hỗ trợ kiểm thử phân tích một chương trình và giám sát sự vận hành của chương trình khi thực thi. Những công cụ chính và chức năng của chúng được trình bày như bảng dưới đây.

Công cụ	Chức năng
Kết xuất bộ nhớ	Đưa ra nội dung của bộ nhớ ngay sau khi chương trình bị ngắt bất thường.
Kết xuất snapshot	Đưa ra nội dung của bộ nhớ trong khi chương trình đang thực thi
Lần vết	Đưa ra các lệnh đã thực thi và nội dung của các thanh ghi ở từng thời điểm.
Tạo dữ liệu kiểm thử	Tự động tạo nhiều loại dữ liệu để kiểm thử
Mô phỏng	Mô phỏng các chức năng không thể thực hiện độc lập; mô phỏng trực tuyến, mô phỏng đơn vị,...
Công cụ khác	Công cụ điều khiển, công cụ chuyển đổi phương tiện, giám định[20] [21]

EUC (Tính toán người dùng cuối)

EUC (End User Computing) là ý tưởng về việc người dùng cuối có thể tự mình thực hiện các qui trình như thiết kế, phát triển, vận hành và bảo trì hệ thống thông tin. Nâng cao năng suất máy tính cá nhân, giảm thiểu chi phí, phát triển xử lí phân tán dựa trên mạng làm việc và phổ biến các gói phần mềm tất cả đều đóng góp cho sự tiến bộ của EUC.

Trước đây, bộ phận người dùng thường yêu cầu rằng việc phát triển và quản lí vận hành phải được thực hiện bởi phòng hệ thống thông tin; tuy nhiên, công việc của các phòng hệ thống thông tin ngày càng tăng và khó có thể cung cấp những dịch vụ riêng biệt cho người dùng cuối. Sau đó, trong nội bộ các bộ phận và các nhóm, người dùng cuối đã bắt đầu tự thực hiện việc vận hành và quản lí hệ thống. Do người dùng cuối tự vận hành, thiết kế, phát triển và tích hợp nên họ có thể điều chỉnh hệ thống phù hợp với những mục đích riêng.

Tích hợp và phát triển hệ thống bởi người dùng cuối được gọi là EUD (End User Development), tuy vậy trong thực tế, sự khác biệt giữa EUC và EUD không được xác định rõ ràng.

Các gói phần mềm

Gói phần mềm là “phần mềm thông dụng mà những người dùng thông thường có thể sử dụng một cách phổ biến”. Trong số nhiều loại gói phần mềm, các gói thương mại nhận được nhiều sự chú ý hơn trong thời gian

[20] **Giám định**: sử dụng bảng hội thoại để thay đổi cưỡng bức dữ liệu và xem nội dung trong quá trình thực thi chương trình.
[21] (FAQ) Các câu hỏi về công cụ hỗ trợ kiểm thử có ở trong các kì thi. Hãy nhớ chính xác các chức năng của kết xuất snapshot và lần vết.

gần đây vì chúng hỗ trợ các qui trình nghiệp vụ một cách hiệu quả.

ERP

ERP (Enterprise Resource Planning – Lập kế hoạch tài nguyên doanh nghiệp) là khái niệm về việc lập kế hoạch cho sự tối ưu hóa trong quản lí tài nguyên nhờ việc hiểu thông tin kinh doanh của toàn công ty.[22] Phần mềm được tích hợp để đạt được mục đích này được gọi là một gói ERP. Gói ERP là gói phần mềm tích hợp tất cả tác vụ chung trong một cơ sở dữ liệu mà không quan tâm đến kiểu tác vụ, chẳng hạn như quản lí sản phẩm, kế toán, quản lí bán hàng, nhân sự và tiền lương.

CRM

CRM (Customer Relationship Management – Quản lí quan hệ khách hàng) là quan niệm trong đó tất cả các phòng ban có cơ hội liên lạc với khách hàng phải chia sẻ, quản lí thông tin cũng như lịch sử liên lạc với khách hàng để bất kì câu hỏi nào từ khách hàng cũng được trả lời thích đáng nhất. Các công ty cố gắng đẩy mạnh việc mở rộng khách hàng của họ bằng việc kết hợp tất cả các kênh truyền thông như điện thoại, fax, Web, e-mail, củng cố mối quan hệ với khách hàng và cung cấp những dịch vụ mà từng khách hàng riêng lẻ cần.

SFA

SFA (Sales Force Automation – Tự động hóa lực lượng bán hàng) là quan niệm cơ bản của hệ thống thông tin giúp ích cho việc tổ chức lại toàn bộ hoạt động bán hàng nhờ đó nâng cao lợi nhuận của tổ chức bằng việc sử dụng công nghệ thông tin. Ví dụ, nhiều hoạt động bán hàng có thể được tạo ra nhờ quản lí các bản ghi liên lạc với mỗi khách hàng trên máy tính. Hơn nữa, việc trao đổi công việc với nhân viên mới cũng được thực hiện tốt hơn.

CTI

CTI (Computer Telephony Integration – Tích hợp điện thoại máy tính) là công nghệ cung cấp các dịch vụ điện thoại cao cấp nhờ việc kết hợp các chức năng xử lí thông tin của máy tính và các chức năng truyền thông của điện thoại.

3.1.5 Các phương pháp phát triển

Điểm chính	➢ Các mô hình qui trình bao gồm mô hình thác đổ, mô hình chế thử và mô hình xoắn ốc. ➢ Các mô hình chi phí bao gồm mô hình COCOMO và mô hình FP.

Mô hình qui trình là mô hình của phương pháp phát triển hệ thống được nhìn nhận theo khía cạnh các qui trình liên quan; **mô hình chi phí** là mô hình theo khía cạnh chi phí.

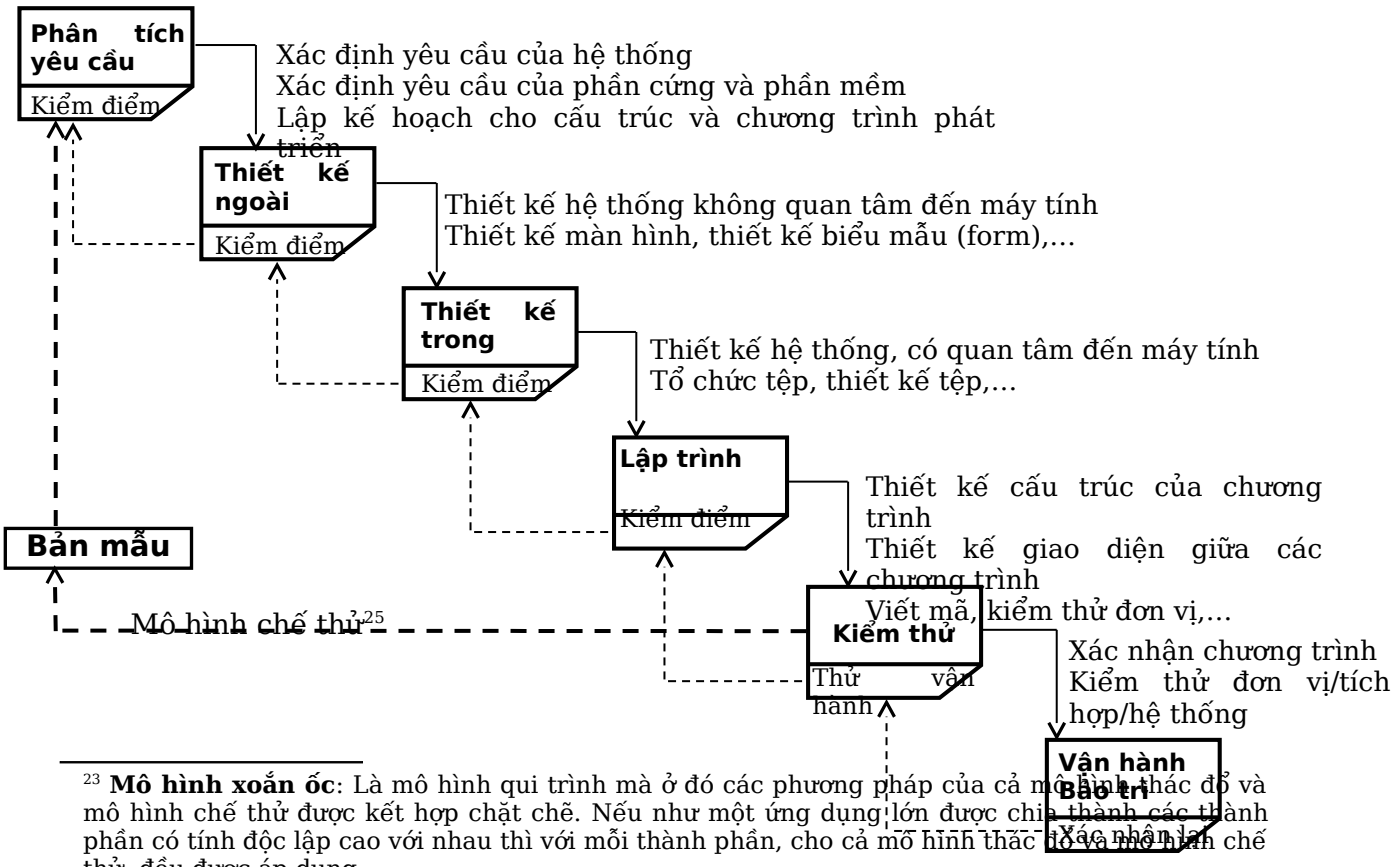
[22] (FAQ) Gần đây, các câu hỏi về các gói phần mềm thường xuất hiện. Đặc biệt, câu hỏi về ERP là nổi bật nhất. Cần phải hiểu được các tính chất của ERP. Để thực hiện ERP, cần xem xét về các qui trình kinh doanh.

Các mô hình qui trình

Mô hình qui trình là mô hình trừu tượng hóa tiến trình phát triển hệ thống. Với việc thiết lập một mô hình qui trình, các thủ tục của phát triển hệ thống sẽ có được sự chỉ đạo hoặc hướng dẫn. Bảng dưới đây mô tả nhiều loại mô hình và các đặc điểm của chúng.

Tên	Đặc điểm
Mô hình thác đổ (Waterfall model)	Mỗi pha của qui trình phát triển sẽ xuôi từ trên xuống dưới (thượng lưu xuống hạ lưu) mà không quay trở lại. • Mỗi pha sẽ được xem xét lại ở thời điểm hoàn thành của nó để phục vụ cho việc quản lí chất lượng. • Khó có thể làm rõ được tất cả các yêu cầu ở thời điểm ban đầu của quá trình phát triển. • Sẽ luôn có những hoạt động cần lặp lại.
Mô hình bản mẫu (Prototyping model)	Một bản mẫu của giao diện người dùng sẽ được phát triển để lọc ra các yêu cầu. • Các yêu cầu được làm rõ trong các giai đoạn sớm. • Giai đoạn cuối cùng sẽ có ít sự hiệu chỉnh và xem xét lại.
Mô hình xoắn ốc[23] (Spiral model)	Các hệ thống con được phát triển một cách độc lập. • Các mức độ phát triển đồng thời có thể được kiểm soát. • Nhân viên phát triển được đảm bảo trong trạng thái ổn định.

Hình dưới đây mô tả ví dụ về các pha phát triển sử dụng mô hình thác đổ và mô hình bản mẫu. Các nội dung về công việc trong qui trình phát triển ở mô hình chế thử hoàn toàn giống như ở mô hình thác đổ. Với giao diện người dùng, việc phân tích yêu cầu và kiểm thử được lặp đi lặp lại cho đến khi các chi tiết hoàn thành. Với những phần khác, mô hình thác đổ được sử dụng. [24]


Phân tích yêu cầu
Kiểm điểm
Xác định yêu cầu của hệ thống
Xác định yêu cầu của phần cứng và phần mềm
Lập kế hoạch cho cấu trúc và chương trình phát triển
Thiết kế ngoài
Kiểm điểm
Thiết kế hệ thống không quan tâm đến máy tính
Thiết kế màn hình, thiết kế biểu mẫu (form),...
Thiết kế trong
Kiểm điểm
Thiết kế hệ thống, có quan tâm đến máy tính
Tổ chức tệp, thiết kế tệp,...
Lập trình
Kiểm điểm
Thiết kế cấu trúc của chương trình
Thiết kế giao diện giữa các chương trình
Viết mã, kiểm thử đơn vị,...
Bản mẫu
Mô hình chế thử[25]
Kiểm thử
Thử vận hành
Xác nhận chương trình
Kiểm thử đơn vị/tích hợp/hệ thống
Vận hành Bảo trì
Xác nhận lại

[23] **Mô hình xoắn ốc**: Là mô hình qui trình mà ở đó các phương pháp của cả mô hình thác đổ và mô hình chế thử được kết hợp chặt chẽ. Nếu như một ứng dụng lớn được chia thành các thành phần có tính độc lập cao với nhau thì với mỗi thành phần, cho cả mô hình thác đổ và mô hình chế thử, đều được áp dụng.

[24] (FAQ) Có nhiều câu hỏi về mô hình thác đổ và mô hình chế thử. Hãy chắc chắn rằng bạn hiểu đúng tất cả các đặc điểm của mỗi mô hình này.

Các mô hình chi phí

Chi phí phần mềm là chi phí phải chịu trong mỗi qui trình của vòng đời phát triển phần mềm (Software Development Life Cycle: SDLC). **Mô hình chi phí** là mô hình định lượng chi phí chẳng hạn như đo lường hiệu quả và chất lượng phần mềm. Các mô hình chi phí và các đặc điểm tương ứng sẽ được trình bày trong bảng dưới đây.

Tên	Đặc điểm
Mô hình COCOMO	Khối lượng công việc lập trình viên phải làm được tính toán trong chi phí dựa trên một công thức toán học, sử dụng một mô hình thống kê và gồm các mức cơ bản, trung bình và nâng cao (chi tiết).
Phương pháp điểm chức năng (Function Point – FP)	Nhóm 5 phần tử - đầu vào, đầu ra, yêu cầu, tệp logic và giao diện – được thu thập và được cộng lại có trọng số. Dựa trên giả sử rằng tổng có trọng số tương quan với các mức phát triển phần mềm, kích thước của việc phát triển sẽ được ước lượng. Cách nhìn nhận theo phương pháp này chính là việc coi thứ người dùng thực sự cần không phải là chương trình mà là chức năng.

[25] (Gợi ý) Vì ý tưởng của mô hình thác đổ rất rõ ràng và dễ hiểu nên có nhiều dự án đã áp dụng phương pháp này. Do các công việc thực hiện theo từng bước một nên mô hình này được sử dụng cho những dự án khá lớn. Mặt khác, mô hình chế thử lại cho thấy sự hiệu quả trong việc phát triển các ứng dụng qui mô nhỏ.

3.1.6 Các phương pháp phân tích yêu cầu

Điểm chính	➢ Biểu đồ DFD và E-R được sử dụng để biểu diễn các kết quả của quá trình phân tích yêu cầu. ➢ Những điều liên quan tới hướng đối tượng bao gồm đóng gói và kế thừa.

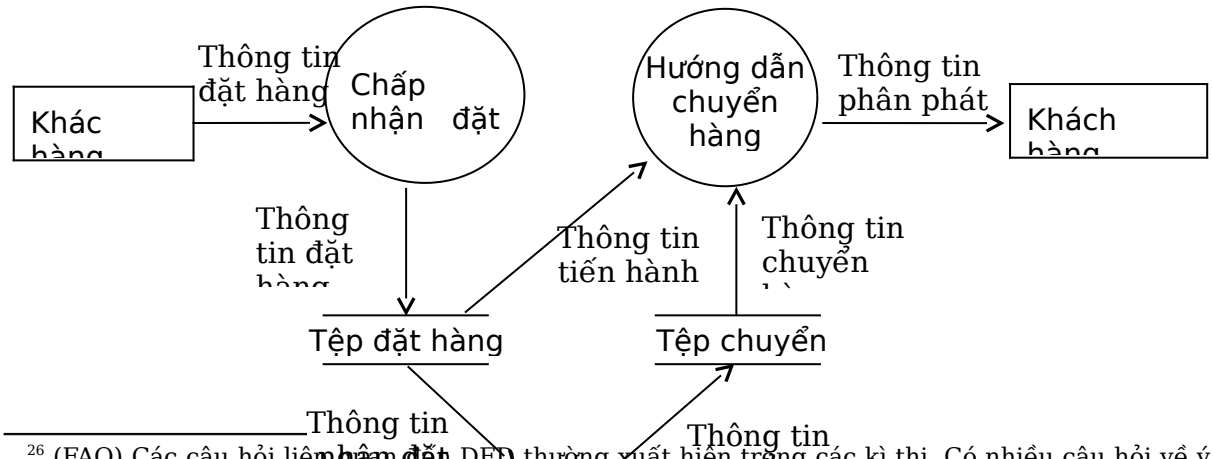
Phân tích yêu cầu là việc nhận ra và tổ chức các yêu cầu của hệ thống một cách cẩn thận. Các kết quả của việc phân tích yêu cầu được mô tả bằng các biểu đồ DFD và E-R. Một phương pháp phân tích khác theo cách nhìn nhận khác là phân tích hướng đối tượng.

DFD (Data Flow Diagram – Biểu đồ luồng dữ liệu)

DFD là một biểu đồ chỉ ra luồng của dữ liệu. Luồng của các vật liệu (các đối tượng) và tiền không nằm trong đó. DFD là cách tiếp cận hướng dữ liệu. Trong DFD, hệ thống được biểu diễn bằng cách sử dụng các kí hiệu trong bảng dưới đây.

Kí hiệu	Tên	Giải thích
▭ ⬭	Thực thể ngoài	Nơi tạo dữ liệu (nguồn), nơi đến (bể chứa, nơi thu)
	Tiến trình	Xử lý dữ liệu: điều chỉnh hoặc chuyển đổi.
→	Luồng dữ liệu	Luồng dữ liệu
=	Kho dữ liệu	Nơi lưu trữ dữ liệu (tệp)

Tiến trình được biểu diễn trong DFD có đặc trưng là không chỉ có luồng dữ liệu vào mà có cả luồng dữ liệu ra để đưa ra kết quả xử lí. Không có tiến trình nào xuất hiện một mình.[26]

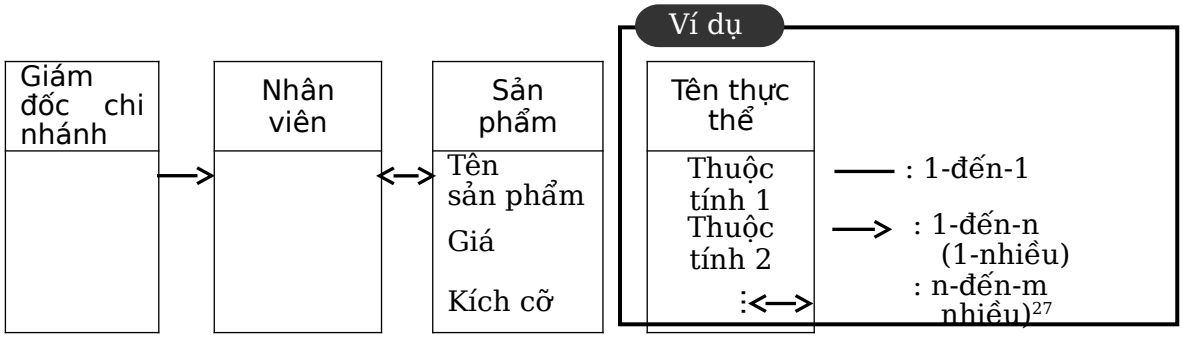


[26] (FAQ) Các câu hỏi liên quan đến DFD thường xuất hiện trong các kì thi. Có nhiều câu hỏi về ý nghĩa của các biểu tượng, vì vậy ít nhất hãy hiểu về ý nghĩa của từng biểu tượng. Những câu hỏi khác bao gồm các câu hỏi liên quan đến cách biểu diễn của DFD. Mỗi tiến trình có đầu vào và đầu ra, vì vậy một biểu đồ luôn luôn có một luồng dữ liệu vào và một luồng dữ liệu ra, giống như "→ ◯ →". Nếu bất kì luồng nào thiếu thì DFD đó có lỗi.

Kiểm kê
hàng

Biểu đồ E-R (Entity-Relationship Diagram – Biểu đồ thực thể liên kết)

Biểu đồ E-R là biểu đồ mô tả **mối quan hệ** giữa **các thực thể**. Một thực thể có thể là người, đối tượng, sự kiện hay khái niệm cần được quản lí trong các qui trình nghiệp vụ, mỗi thực thể đều có các thuộc tính. Dưới đây là một ví dụ về biểu đồ E-R. Chú ý là bỏ qua các thuộc tính của giám đốc chi nhánh và các nhân viên.



Biểu đồ trên chỉ ra một giám đốc chi nhánh (1) quản lí một số nhân viên (n). Mỗi nhân viên lại phụ trách một số (m) sản phẩm và mỗi sản phẩm có thể được phụ trách bởi một số (n) nhân viên. Mỗi sản phẩm có các thuộc tính là tên sản phẩm, giá và kích cỡ.

Hướng đối tượng

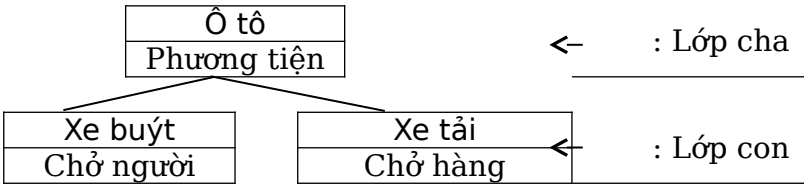
Hướng đối tượng tức là mô hình hóa dữ liệu và các thao tác (tiến trình) cùng nhau. Việc tích hợp dữ liệu và thao tác được gọi là **đóng gói**. Các thuộc tính được chia sẻ bởi dữ liệu sẽ được trích xuất ra, và dữ liệu được tổ chức thành các lớp trong một cấu trúc phân cấp. Các lớp mức thấp hơn kế thừa các thuộc tính của các lớp mức cao hơn trong cấu trúc này, và các thuộc tính được kế thừa theo cách như vậy gọi là "kế thừa."[28]

Ví dụ, ta quan tâm đến mối quan hệ giữa ô tô và xe buýt. Nếu ô tô được định nghĩa như một "phương tiện," và xe buýt được định nghĩa như một "phương tiện chở người," thì thuộc tính "là phương tiện" sẽ là chung của cả hai loại này, vì vậy ô tô là lớp cha trong khi xe buýt là lớp con. Khi ấy, sử dụng chức năng kế thừa, xe buýt có thể được định nghĩa đơn giản là "chở người."[29]

[27] (Chú ý) Mối quan hệ giữa các thực thể được gọi là sự tương ứng. Mối quan hệ giữa giám đốc chi nhánh và nhân viên là 1-đến-n (1-nhiều), và mối quan hệ giữa nhân viên với sản phẩm là n-đến-m (nhiều-nhiều). Mối quan hệ giữa giám đốc chi nhánh và nhân viên chỉ ra rằng mỗi nhân viên chỉ có một giám đốc chi nhánh nhưng mỗi giám đốc chi nhánh có thể có nhiều nhân viên. Nếu một nhân viên được chọn, giám đốc chi nhánh của nhân viên đó là duy nhất, nhưng khi chọn một giám đốc chi nhánh thì không phải có duy nhất một nhân viên liên quan đến ông ta. Ngược lại, trong mối quan hệ giữa nhân viên và sản phẩm, không thể chọn duy nhất một sản phẩm theo một nhân viên cũng như không thể chọn duy nhất một nhân viên phụ trách một sản phẩm.

[28] (Gợi ý) Trong hướng đối tượng, ta chỉ cần thiết kế các phần được thêm vào. Chẳng hạn, để thêm lớp xe tải, ta chỉ cần thêm phần "chở hàng". Tuy nhiên, trong thực tế, khó có thể nhận biết được những thuộc tính chung và những thuộc tính được thêm vào.

[29] **Lớp cha/Lớp con**: Lớp ở cấp cao hơn được gọi là lớp cha (superclass) còn lớp ở cấp thấp hơn được gọi là lớp con (subclass)..



3.1.7 Quản lí chất lượng phần mềm

Điểm chính	➢ Việc kiểm điểm bao gồm "duyệt qua" (walk-through) và giám định (inspection). ➢ Tỉ lệ phát hiện lỗi của phần mềm giống như đường cong tăng trưởng.

Quản lí chất lượng phần mềm là nói đến quá trình đánh giá và quản lí chất lượng phần mềm nhằm thỏa mãn các yêu cầu của người dùng. Các phương pháp cho việc quản lí này bao gồm kiểm điểm (review), dự báo độ tin cậy,...

Các phương pháp kiểm điểm

Kiểm điểm là việc họp thảo luận được tiến hành ở cuối mỗi tiến trình để tránh mang những vấn đề đang tồn tại vào tiến trình tiếp theo trong quá trình phát triển hệ thống.

Những điểm cần chú ý của một buổi kiểm điểm là:

- Nên có từ 4 đến 6 thành viên (nếu có quá nhiều, sẽ không thể có được sự nhất trí).
- Tài liệu cần được phân phát trước. (Các vấn đề nên được liệt kê ra trước.)
- Mục đích là tìm lỗi. (Các biện pháp để loại trừ chúng sẽ được thảo luận sau.)
- Buổi họp nên giới hạn trong 1 đến 2 giờ (nếu cần họp lâu hơn thì sẽ họp thêm vào ngày khác).
- Ban quản lí không nên tham dự buổi họp (điều này có thể dẫn đến những đánh giá mang tính cá nhân).

Có nhiều loại kiểm điểm, tùy thuộc vào mức áp dụng:[30]

Loại	Chức năng
Kiểm điểm thiết kế (Design	Áp dụng cho mỗi tiến trình thiết kế (thiết kế ngoài, thiết kế trong, thiết kế chương trình) của phát triển hệ thống. Được dùng cho việc đánh giá các văn bản thiết kế và xác nhận các

[30] (FAQ) Ý nghĩa của thuật ngữ "kiểm điểm" và sự khác nhau giữa duyệt qua và giám định thường được hỏi trong kì thi.
[31] **Người điều tiết**: người điều tiết là người quản lý được đào tạo để sắp đặt, chỉ đạo các buổi kiểm điểm và có thể xử lý các lỗi đã phát hiện. Người điều tiết chọn các kiểm điểm viênđược gọi là thanh tra viênlà những người có khả năng và kinh nghiệm đẻ thẩm định các sản phẩm bàn giao ở mỗi tiến trình.
[32] **Giám định mã:** Giám định mã là việc giám định trong chương trình nguồn. Trong khi giám định mã, chương trình nguồn được kiểm tra và thông qua trên cơ sở từng dòng lệnh.

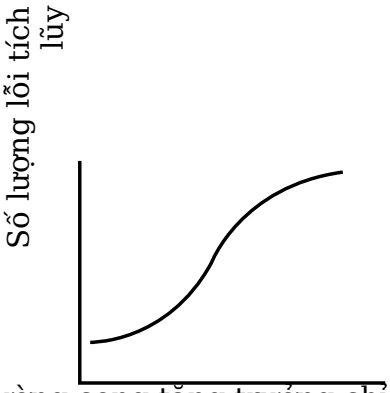
review)	giao diện,…
Duyệt qua (Walk-through)	Dùng cho tất cả các tiến trình của phát triển hệ thống. Trong các pha sớm, không chỉ đội phát triển mà cả người dùng cũng tham gia vào đó.
Giám định (Inspection)	Dùng cho tất cả các tiến trình của phát triển hệ thống. Nó được thực hiện thực hiện một cách có hệ thống dưới sự hướng dẫn của người điều tiết.[31] Những vấn đề đưa ra nên được thông báo tới toàn bộ dự án. Riêng việc giám định ở trong pha lập trình được gọi là giám định mã.[32]

[31] **Người điều tiết**: người điều tiết là người quản lí được đào tạo để sắp đặt, chỉ đạo các buổi kiểm điểm và có thể xử lý các lỗi đã phát hiện. Người điều tiết chọn các kiểm điểm viên được gọi là thanh tra viên là những người có khả năng và kinh nghiệm để thẩm định các sản phẩm bàn giao ở mỗi tiến trình.

[32] **Duyệt mã**: Duyệt mã là việc xét duyệt trong chương trình nguồn. Trong khi duyệt mã, chương trình nguồn được kiểm tra và thông qua trên cơ sở từng dòng lệnh.

Đường cong tăng trưởng

Trong pha kiểm thử, mối quan hệ giữa số lượng tích lũy của các lỗi đã tìm ra và thời gian được coi như giống với đường cong tăng trưởng. Các tính chất của đường cong tăng trưởng được trình bày dưới đây. Đường cong tăng trưởng đôi khi còn được gọi là **đường cong hình chữ S (S-shape curve)**.



Ở giai đoạn đầu, tăng từ từ.
Số lượng lỗi tăng đột biến ở một thời điểm cụ thể.
Cuối cùng, nó đạt đến trạng thái bão hòa.

Đường cong tăng trưởng chỉ ra rằng các lỗi không dễ dàng phát hiện ở thời gian đầu, do vậy sau thời gian này số lượng lỗi sẽ tăng nhanh và sẽ giảm dần ở phần cuối (một cách lí tưởng).[33] [34]

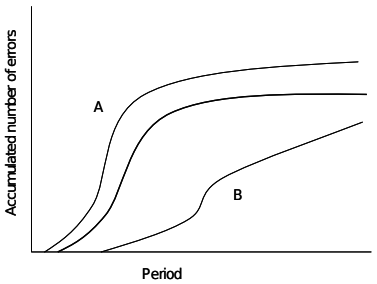
Mô hình gieo lỗi (Error-Planting Model)

Mô hình gieo lỗi còn được gọi là **mô hình phát tán lỗi** (error-spreading model) hay **mô hình gắn lỗi** (bug-embedding model). Ở mô hình này, các lỗi sẽ được cố tình đặt vào chương trình. Khi ấy tỉ lệ giữa số lỗi được gieo và số lỗi được tìm ra sẽ được dùng trong việc phân phối lỗi cho phù hợp để ước lượng được tổng số lỗi của chương trình. Ngày nay, mô hình này đã được cải tiến không đưa lỗi vào chương trình nữa. Thay vào đó, hai nhóm kiểm thử độc lập sẽ thực hiện trên cùng một chương trình, và số lượng lỗi được tìm ra bởi mỗi nhóm sẽ được dùng để ước lượng tổng số lỗi.

---

[33] (FAQ) Có các câu hỏi lựa chọn đường cong tăng trưởng đúng. Ví dụ, có thể có vài đồ thị trong nhóm các câu trả lời, và câu hỏi có thể là: "Đường cong nào trong số các đường cong dưới đây minh họa việc kiểm thử được thực hiện theo đúng kế hoạch?" Hãy nắm rõ các đặc điểm của đường cong hình chữ S.

[34] (Gợi ý)



Đồ thị "A" chỉ ra rằng số lượng lỗi lớn hơn đường cong lỗi chuẩn. Ta có thể hiểu rằng dữ liệu kiểm thử rất tốt nên có nhiều lỗi được tìm ra. Tuy nhiên, cũng có thể thấy rằng lỗi được tìm ra từ rất sớm vì chất lượng của phần mềm là kém. Ở đồ thị "B", số lỗi tích lũy không ổn định, vì vậy có thể thấy chất lượng phần mềm là rất kém.

Các đặc trưng chất lượng phần mềm

Các đặc trưng chất lượng phần mềm là các chuẩn mà theo đó chất lượng của phần mềm được đánh giá. ISO/IEC9126-1 gồm các chuẩn quốc tế của các đặc trưng chất lượng phần mềm, nó gồm 6 đặc trưng được liệt kê trong bảng dưới đây.

Các đặc trưng chất lượng	Định nghĩa
Chức năng	Các chức năng và mục đích tương xứng với nhau
Độ tin cậy	Các chức năng xác định làm việc dưới các điều kiện xác định và việc phục hồi lại sau khi có lỗi là dễ dàng.
Tính tiện lợi	Mục đích và chức năng rõ ràng, vận hành dễ dàng.
Hiệu quả	Thời gian thực thi nhanh, sử dụng các tài nguyên một cách hiệu quả.
Khả năng bảo trì	Thay đổi và sửa chữa dễ dàng.
Khả năng di chuyển	Dễ dàng chuyển sang môi trường khác.

Câu hỏi nhanh

Q1 Liệt kê các ngôn ngữ lập trình được phân loại là “thủ tục”, “chức năng”, “logic” và “hướng đối tượng”.

Q2 Giải thích “đồng sử dụng”.

Q3 Các loại bộ xử lí ngôn ngữ?

Q4 Liệt kê 3 mô hình qui trình điển hình và giải thích các đặc điểm của chúng.

Q5 Liệt kê hai sự khác biệt giữa giám định (inspoction) và duyệt qua (walk-through).

A1 Thủ tục: COBOL, C, Fortran, và Pascal
Chức năng: Lisp
Logic: Prolog
Hướng đối tượng: Java, C++, and Smalltalk

A2 Thuật ngữ này nói đến cấu trúc chương trình ở đó một chương trình có thể được sử dụng bởi nhiều tác vụ ở cùng một thời điểm; đó là một kiểu của cấu trúc chương trình “dùng lại được”, cấu trúc chương trình này cho phép nhiều chương trình (tác vụ) chia sẻ việc sử dụng chương trình mà không phải nạp lại chương trình đó vào bộ nhớ chính.

A3 Trình biên dịch (compiler), trình dịch hợp ngữ (assembler), trình sinh mã (generator), và trình thông dịch (interpreter)

A4 Mô hình thác đổ: Mỗi pha của qui trình phát triển sẽ đi từ trên xuống dưới (thượng lưu xuống hạ lưu) mà không quay lại.
Mô hình chế thử: Một bản mẫu của giao diện người dùng sẽ được phát triển để làm rõ yêu cầu.
Mô hình xoắn ốc: Các hệ thống con được phát triển độc lập.

A5 Cả giám định và duyệt qua đều là các phương pháp kiểm điểm và tương tự nhau; tuy nhiên, chúng khác nhau ở phương thức vận hành và các tiến trình phải theo:
Giám định là thực hiện dưới sự hướng dẫn của người điều tiết, đó là người được đào tạo để sắp đặt và chỉ đạo các buổi kiểm điểm. Các lỗi được tìm ra trong giám định được thông báo tới toàn bộ dự án.

3.2 Các nhiệm vụ của qui trình phát triển hệ thống

Mở đầu

Phương pháp điển hình nhất trong phát triển hệ thống là mô hình thác đổ. Ở đây, chúng ta sẽ theo các pha của mô hình thác đổ và tổ chức nội dung của các hoạt động của từng pha.

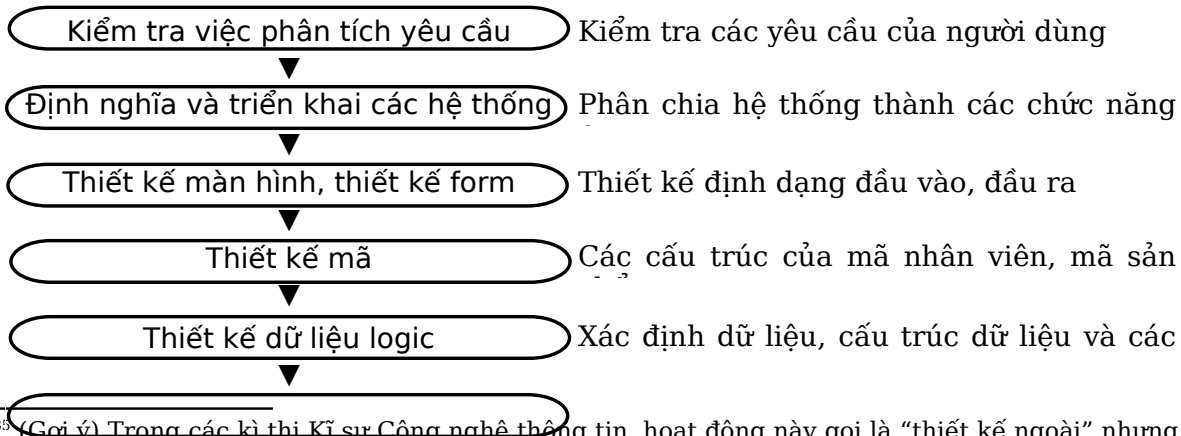
3.2.1 Thiết kế ngoài

Điểm chính	➢ Thiết kế ngoài bao gồm thiết kế màn hình, thiết kế biểu mẫu và thiết kế mã. ➢ Trong thiết kế mã, khả năng mở rộng là cần thiết và tính dễ học là rất quan trọng.

Thiết kế ngoài nói đến việc thiết kế hệ thống mà không quan tâm đến máy tính. Kết quả của các hoạt động sẽ được tổng hợp lại trong các đặc tả thiết kế ngoài để phục vụ cho việc kiểm điểm.[35]

Thủ tục của thiết kế ngoài

Thiết kế ngoài là các hoạt động thu thập và phân tích các yêu cầu hệ thống được đưa ra bởi người dùng và thiết lập các chức năng hệ thống dựa trên những kết quả trước đó. Do không quan tâm đến các đặc tả của máy tính, thiết kế được thực hiện theo các chức năng ứng dụng dưới cách nhìn nhận theo quan điểm của người dùng. Ở đây, các hoạt động chính bao gồm thiết kế màn hình, thiết kế biểu mẫu và thiết kế mã. Luồng thiết kế ngoài được mô tả dưới đây.


Kiểm tra việc phân tích yêu cầu
Kiểm tra các yêu cầu của người dùng
Định nghĩa và triển khai các hệ thống
Phân chia hệ thống thành các chức năng
Thiết kế màn hình, thiết kế form
Thiết kế định dạng đầu vào, đầu ra
Thiết kế mã
Các cấu trúc của mã nhân viên, mã sản
Thiết kế dữ liệu logic
Xác định dữ liệu, cấu trúc dữ liệu và các

[35] (Gợi ý) Trong các kì thi Kĩ sư Công nghệ thông tin, hoạt động này gọi là "thiết kế ngoài" nhưng một số sách tham khảo có thể gọi là "thiết kế chức năng", "thiết kế tổng thể" hay "thiết kế phác thảo". Nội dung của các hoạt động này giống nhau.

Chuẩn bị các đặc tả thiết kế ngoài Tổng hợp

Những điểm cần nhớ trong thiết kế màn hình

Trong thiết kế màn hình, điều quan trọng là phải thiết kế với suy nghĩ là tạo ra giao diện phù hợp với người dùng (giao diện con người). Do đó, cần phải quan tâm đến việc đơn giản hóa đầu vào và có một màn hình dễ thao tác. Chi tiết hơn, những điểm dưới đây cần phải quan tâm tới:[36]

- Quan tâm đến nguồn dữ liệu, lượng dữ liệu, số phần tử và các con số, thuộc tính,…
- Đặt các phần tử đầu vào sao cho chúng được đặt từ trên xuống dưới, từ trái qua phải.
- Cố gắng chuẩn hóa bố cục và thao tác màn hình.
- Giữ được tính nhất quán trong việc biểu diễn thông điệp.
- Quan tâm đến các khả năng bỏ dở thao tác giữa chừng hoặc khởi động lại từ màn hình trước.

Những điểm cần nhớ trong thiết kế đầu ra

Thiết kế đầu ra là việc thiết kế định dạng đầu ra của hệ thống, bao gồm việc hiển thị màn hình và in ấn. Trong đó, in ấn (báo cáo) là trung tâm. Với thiết kế màn hình, cần phải quan tâm đến giao diện người dùng và làm cho các bản báo cáo dễ đọc. Chi tiết hơn, những điểm dưới đây cần phải quan tâm tới:

- Thiết lập thứ tự và các vị trí có quan tâm đến mối liên hệ giữa các mục.
- Đảm bảo rằng tiêu đề được biểu diễn thích hợp với nội dung được in ra.
- Phân biệt rõ ràng các ngày khác nhau như ngày chuẩn bị, ngày báo cáo và ngày phê chuẩn.
- Đặt các mục với khoảng cách thích hợp.
- Bố trí toàn bộ báo cáo để có đủ không gian trống.
- Quan tâm đến thiết kế sao cho các phần tử chủ chốt cần được nhận ra ngay.

Thiết kế mã

Mã cần có các chức năng được liệt kê trong bảng dưới đây. Khi thiết kế, ta phải quan tâm đến nhiều thuộc tính như tính phổ biến, tính hệ thống, tính mở rộng và tính rõ ràng.[37]

Chức năng	Giải thích
Nhận dạng	Là chức năng phân biệt dữ liệu. Mã có thể phân biệt hai người có cùng họ và tên.
Phân loại	Là chức năng gom nhóm; phân loại bởi việc sáp nhập mã,…
Liệt kê	Là chức năng sắp xếp: Nếu các con số được canh lề thì dữ liệu

[36] (FAQ) Kiểu câu hỏi sau thường được hỏi trong các kì thi: “Đâu là mô tả thích hợp cần nhớ trong thiết kế màn hình và thiết kế biểu mẫu?” Hãy đọc kĩ các mô tả để trả lời chúng.

[37] (Chú ý) Các ví dụ về mã bao gồm mã liên tục và mã có các mã phân lớp. Mã liên tục là các số liên tục được gán cho dữ liệu theo thứ tự từ đầu. Nó có ích khi lượng dữ liệu là cố định. Trong mã có các mã phân lớp, dữ liệu được phân loại thành các lớp lớn, lớp trung bình và lớp nhỏ với những tiêu chuẩn cụ thể trong cấu trúc phân cấp, và mỗi nhóm lại có các mã liên tục. Các con số có thể dài, nhưng chúng phù hợp cho việc xử lí náy tính. Hệ thống mã bưu điện là một ví dụ về loại mã này.

	cần được sắp xếp theo ngày sinh.
Kiểm tra	Là chức năng kiểm tra giá trị đầu vào; ví dụ, bổ sung thêm số kiểm tra.[38]

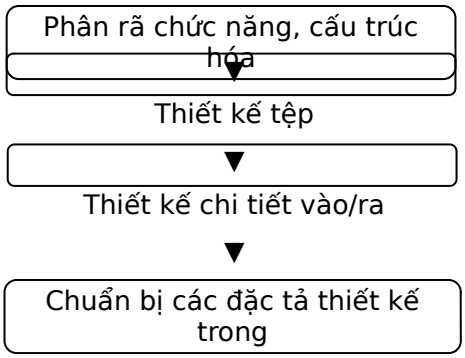
3.2.2 Thiết kế trong

Điểm chính	➢ Thiết kế trong bao gồm phân rã chức năng, cấu trúc hóa, thiết kế tệp, và thiết kế chi tiết vào/ra. ➢ Thiết kế chi tiết vào/ra có trọng tâm là phương pháp kiểm tra dữ liệu đầu vào.

Thiết kế trong là việc thiết kế hệ thống có quan tâm đến máy tính nào sẽ được sử dụng. Kết quả của các hoạt động này sẽ được tổng hợp trong báo cáo thiết kế trong để phục vụ cho việc kiểm điểm.

Thủ tục của thiết kế trong

Đứng trên quan điểm của người phát triển hệ thống, một hệ thống tối ưu sẽ được thiết kế dựa trên các đặc tả về máy tính. Luồng thiết kết trong được trình bày dưới đây.



Xác định các chức năng và gom nhóm lại dựa vào nội dung xử lí.

Quyết định một cách cụ thể về phương tiện, cách tổ chức, bố cục,... của tệp

Quyết định phương tiện, phương thức, phương pháp kiểm tra đầu vào/đầu ra.

Tổng hợp

Thiết kế tệp

Tổ chức và phương tiện của các tệp cần phải được quyết định theo mục đích sử dụng của chúng. Để phục vụ cho việc sao lưu, nhiều loại phương tiện được sử dụng, bao gồm các đĩa từ (MO), đĩa mềm (FD), CD-R, CD-RW, DVD_R, DVD-RAM, DAT,...[39]

[38] **Kiểm tra số**: Mã một chữ số (1-digit) có được nhờ thực hiện một tính toán nào đó đã được xác định trước trên mỗi số của mục số. Khi một mã được nhập vào, một tính toán như thế được thực hiện để thu được số có một chữ số, sau đó số này sẽ được so sánh với số kiểm tra. Nếu chúng giống nhau, mã được coi là hợp lệ; ngược lại, nó được coi là không hợp lệ. Đây là một phương pháp phát hiện lỗi.

[39] (Gợi ý) Một phương tiện thích hợp được lựa chọn tùy thuộc vào cái gì sẽ được sao lưu. Dung lượng gần đúng của mỗi phương tiện được trình bày dưới đây:

Tùy theo mục đích sử dụng, phương pháp tổ chức tệp sẽ được chọn từ những phương án bao gồm tuần tự, trực tiếp, tuần tự có đánh chỉ mục và phân vùng.[40] Nếu lượng lớn các bản ghi tồn tại và hầu hết trong số chúng được đọc và cập nhật thì cách tổ chức tuần tự là phù hợp. Còn với việc xử lí ngẫu nhiên, cách tổ chức trực tiếp lại phù hợp hơn.

Các phương pháp kiểm tra

Trong số các phương pháp được dùng cho việc kiểm tra dữ liệu đầu vào, một số phương pháp điển hình được trình bày trong bảng dưới đây.

Phương pháp	Nội dung kiểm tra
Kiểm tra số	Phần tử số có thực sự là một số không?
Kiểm tra định dạng	Nó có nằm trong định dạng định xác định trước hay không? Các số có không được canh lề không?
Kiểm tra giới hạn	Giá trị có nằm trong khoảng giới hạn trên và giới hạn dưới hay không?
Kiểm tra khoảng	Giá trị có nằm trong khoảng đúng không? (Đây có thể được coi như một kiểu kiểm tra giới hạn)
Kiểm tra hợp lệ	Giá trị có hợp lệ không? (Ví dụ, ngày 29 tháng 2 chỉ có ở năm nhuận)
Kiểm tra thứ tự	Các giá trị khóa có được đặt theo thứ tự không?
Kiểm tra cân bằng	Các mục theo cặp có tương xứng không? (Ví dụ, người thuê nhà và chủ nhà)
Kiểm tra phân loại	Giá trị mã có nằm trong tệp chủ (master file) hay không?

Phương pháp số kiểm tra

Đây là phương pháp xác định liệu có lỗi đầu vào hay không khi một mã được nhập vào bằng việc thực hiện thực hiện tính toán giống như khi mã được tạo ra và so sánh kết quả tính toán với mã. Thông thường, một chữ số (số kiểm tra) sẽ được thêm vào cuối mã cơ sở để tạo ra mã.

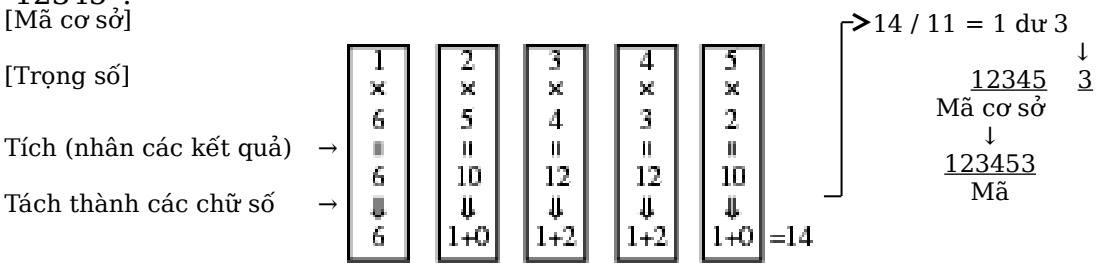
XXXXX	Y	→	XXXXXY
Mã cơ sở	Số kiểm tra		Mã

Phương pháp số kiểm tra thường sử dụng *modulus (chia lấy dư) 11*. Trong *modulus 11*, các trọng số 2, 3,... được gán vào mỗi chữ số của mã cơ sở từ chữ số nhỏ nhất đầu cùng bên phải. Tích của trọng số và giá trị số của mỗi

MO: 128, 320, 540, 640 MB/ 1.3 GB
FD: 1.2/ 1.44 MB
CD-R: 640/700 MB
CD-RW: 700 MB
DVD-R: 4.7/ 8.5 GB
DVD+R: 4.7/8.5 GB
DVD-RAM: 4.7 GB
DVD-RW: 4.7 GB
DVD+RW: 4.7 GB
DAT: Tối đa 24 GB

[40] (Gợi ý) Tổ chức phân vùng ít khi được sử dụng trong các tệp dữ liệu. Nó hầu như luôn được sử dụng trong các tệp thư viện.

chữ số sau đó được tính toán và tính tổng các tích. Cuối cùng, tổng này được chia cho 11, và số dư chính là số kiểm tra. Nếu một tích thu được là số có 2 chữ số thì các chữ số này sẽ được tách ra. Dưới đây là ví dụ với mã cơ sở "12345".


[Mã cơ sở]
[Trọng số]
Tích (nhân các kết quả) →
Tách thành các chữ số →
1 2 3 4 5
× × × × ×
6 5 4 3 2
= = = = =
6 10 12 12 10
⇓ ⇓ ⇓ ⇓ ⇓
6 1+0 1+2 1+2 1+0 =14
14 / 11 = 1 dư 3
12345 3
Mã cơ sở
123453
Mã

Vì chia cho 11 nên phương pháp này được gọi là *modulus 11*. Chia một số cho 11 có thể dư 10. Trong trường hợp này, số kiểm tra được định nghĩa là 0. Bên cạnh đó, cũng có một phương pháp được gọi là *modulus 10.*[41]

[41] **Modulus 10:** Ý tưởng về việc tính toán số kiểm tra bởi phương pháp *modulus 10* hoàn toàn giống với ý tưởng của phương pháp *modulus 11* ngoại trừ việc tổng có trọng số được chia cho 10 thay vì 11. Cả phương pháp *modulus 11* và *modulus 10* đều có thể phát hiện ra hầu hết các lỗi đầu vào.

3.2.3 Các phương pháp thiết kế phần mềm

Điểm chính	➢ Các phương pháp thiết kế phần mềm bao gồm thiết kế có cấu trúc và thiết kế theo cấu trúc dữ liệu. ➢ Thiết kế có cấu trúc là phương pháp thiết kế hướng qui trình.

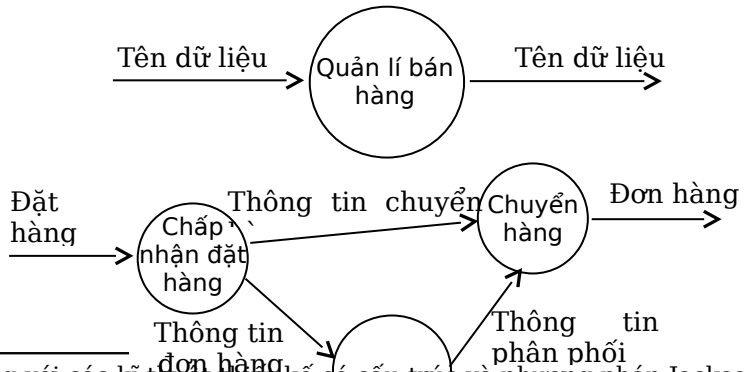
Trong khi phần mềm được mô hình hóa thành các qui trình và dữ liệu một cách riêng rẽ thì thiết kế có cấu trúc lại là kĩ thuật hướng qui trình. Ngoài ra, có một kĩ thuật khác mà ở đó các hoạt động thiết kế được thực hiện với sự tập trung vào cấu trúc dữ liệu. Các kĩ thuật thiết kế có cấu trúc bao gồm các biểu đồ bong bóng, phương pháp phân rã STS, và phương pháp phân rã TR. Các kĩ thuật tập trung vào cấu trúc dữ liệu bao gồm phương pháp Jackson và phương pháp Warnier.[42]

Các kĩ thuật thiết kế có cấu trúc

Thiết kế có cấu trúc là kĩ thuật thiết kế dựa trên cách tiếp cận của việc cấu trúc hóa. Biểu đồ cấu trúc được sử dụng cho phương pháp này. Trong mô hình này, các hoạt động thiết kế thực hiện từ những ý tưởng tổng quát đến chi tiết, vì vậy có khi nó được gọi là "lọc mịn từng bước" (cách tiếp cận trên-xuống (top-down) hay phân rã mô-đun[43]).

Biểu đồ bong bóng (Bubble chart)

Những biểu đồ này sử dụng các bong bong (các hình tròn) để biểu diễn các quá trình chuyển đổi dữ liệu đầu vào thành dữ liệu đầu ra, chúng tương tự như DFD. Với mọi hệ thống, chỉ có một bong bóng khởi đầu, nhưng cùng với quá trình làm vỡ (cấu trúc hóa lại) thì các bong bóng tạo thành và các luồng dữ liệu ngày một phức tạp hơn. Dưới đây là một ví dụ về việc phân rã hệ thống quản lí bán hàng.



[42] (FAQ) Cùng với các kĩ thuật thiết kế có cấu trúc và phương pháp Jackson, các câu hỏi bao trùm tổng quan cùa các phương pháp thiết kế phần mềm cũng thường xuất hiện; chúng bao gồm các câu hỏi như "Lựa chọn nào dưới đây là một ví dụ về…?" và nhóm câu trả lời thường liệt kê ra các thuật ngữ.

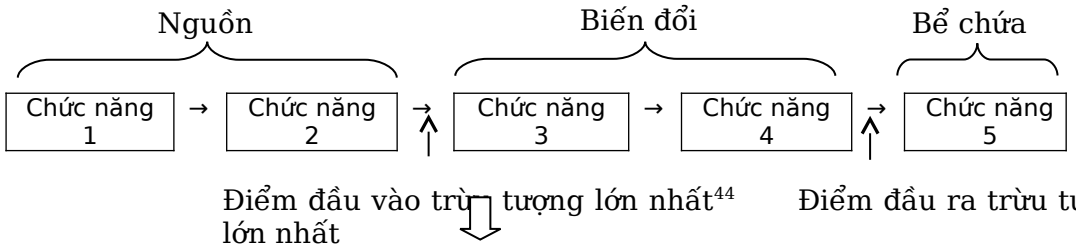
[43] **Phân rã mô-đun**: Đây là qui trình trong đó các chức năng chương trình được thảo luận, được chia thành các đơn vị chức năng, và đặt vào trong một phân cấp chức năng. Mỗi đơn vị chức năng sau khi được phân rã được gọi là một mô-đun. Do đó một chương trình bao gồm nhiều mô-đun.

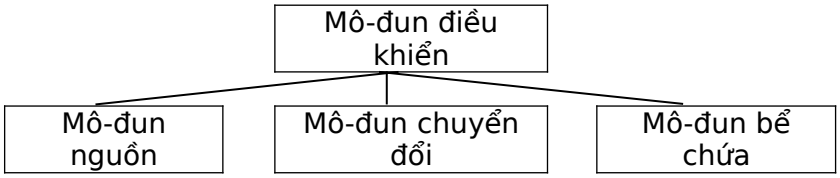
Phân phối
đơn hàng

Giống như DFD, biểu đồ bong bóng được sử dụng để phân rã các chức năng của hệ thống. Nó cũng được dùng để phân rã các chức năng của chương trình để tạo nên một sự phân cấp.

Phân rã STS (source, transform, sink – nguồn, biến đổi, bể chứa)

Trong kĩ thuật này, cấu trúc chương trình được chia thành nguồn (đầu vào), biến đổi (các tiến trình) và bể chứa (đầu ra), và mỗi thành phần được định nghĩa như một chức năng. Một khi phân rã STS được thực hiện thì các kĩ thuật khác như DFD hay biểu đồ bong bóng cũng được sử dụng.

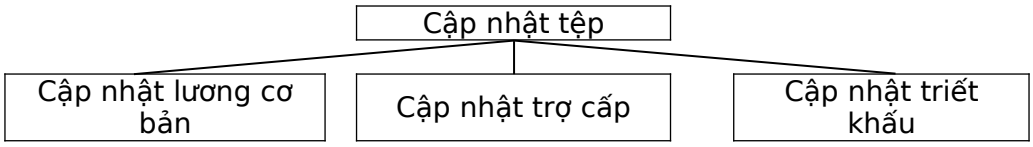




Trong phân rã STS, sau khi chia thành 3 mô-đun, ta bổ sung thêm một *mô-đun điều khiển* để điều khiển những mô-đun đó. Phương pháp này được dùng khi viết chương trình cho xử lí theo lô.

Phân rã TR (Transaction Partitioning – Phân rã giao dịch)

Trong kĩ thuật này, các giao dịch được phân rã bởi chức năng đối với luồng của dữ liệu, và chúng được định dạng thành các mô-đun. Ví dụ dưới đây là một phần của chương trình tiền lương. Nếu chức năng "cập nhật tệp" có 3 chức năng – cập nhật lương cơ bản, cập nhật tiền trợ cấp (tiền lương), và cập nhật triết khấu – thì mỗi chức năng tạo thành một mô-đun:



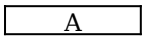
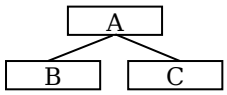
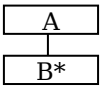
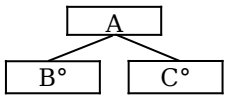
Phân rã TR được sử dụng thường xuyên trong các chương trình của các hệ thống trực tuyến. Ví dụ, trong một hệ thống đặt chỗ, các giao dịch xảy ra thường xuyên theo mỗi chức năng, chẳng hạn như "yêu cầu thông tin về số lượng chỗ còn trống" hoặc "xuất vé".

[44] Điểm trừu tượng lớn nhất: Đây là khái niệm trong phân rã STS mà ở đó chương trình được phân rã thành 3 mô-đun. Ranh giới giữa nguồn và biến đổi được gọi là điểm đầu vào trừu tượng lớn nhất, ranh giới giữa biến đổi và bể chứa được gọi là điểm đầu ra trừu tượng lớn nhất. Điểm đầu vào trừu tượng lớn nhất là nơi dữ liệu đầu vào được trừu tượng hóa nhiều nhất, ở đó dữ liệu được chuyển đổi và dừng lại trở thành dữ liệu đầu vào. Điểm đầu ra trừu tượng lớn nhất là nơi dữ liệu đầu ra được trừu tượng hóa nhiều nhất, ở đó dữ liệu được tổ chức lại để hình thành dữ liệu đầu ra.

Các kĩ thuật thiết kế dựa trên cấu trúc dữ liệu

Với ý tưởng là việc làm rõ bản chất cấu trúc dữ liệu vào/ra sẽ xác định các tiến trình, những kĩ thuật này sẽ so sánh cấu trúc dữ liệu đầu vào với cấu trúc dữ liệu đầu ra để đưa ra cấu trúc của các tiến trình. Một phương pháp điển hình là phương pháp Jackson (Jackson Structured Programming: JSP). Trong phương pháp Jackson, cấu trúc dữ liệu và cấu trúc chương trình được biểu diễn như cấu trúc cây.

Bảng dưới đâu mô tả các kí hiệu được sử dụng trong phương pháp Jackson và ý nghĩa của chúng.

Kí hiệu	Tên	Ý nghĩa
	Phần tử cơ bản	Một phần tử trong cấu trúc dữ liệu; một đơn vị xử lí trong cấu trúc chương trình.
	Trình tự	A gồm có B và C; B và C được thực thi theo thứ tự đó. Mỗi phần tử chỉ xuất hiện một lần trong thứ tự đó.
	Lặp lại	Bên trong A; B được lặp lại ít nhất 0 lần (có thể không lặp). Kí hiệu “*” có ý nghĩa là sự lặp lại.
	Lựa chọn	Cả B và C đều được chọn bởi A (mỗi thời điểm chỉ có một phần tử).

Phương pháp Jackson được sử dụng thường xuyên trong các chương trình của các hệ thống xử lí thương mại. Một phương pháp khác cũng dựa trên cấu trúc dữ liệu là phương pháp Warnier.[45]

[45] **Phương pháp Warnier**: Giống như phương pháp Jackson, đây là kĩ thuật thiết kế dựa trên cấu trúc dữ liệu. Nó được mô tả bằng cách vẽ biểu đồ Warnier, tương tự như lưu đồ.

3.2.4 Tiêu chuẩn phân rã mô-đun

Điểm chính	➢ Độ độc lập mô-đun được sử dụng cho các tiêu chuẩn phân rã mô-đun. ➢ Độ bền và độ móc nối (cố kết) mô-đun được sử dụng cho các tiêu chuẩn của độ độc lập mô-đun.

Trong thiết kế có cấu trúc, tất cả mô-đun được phân rã cuối cùng sẽ được đánh giá dựa trên nhiều tiêu chí đánh giá như cấu trúc hoặc độ độc lập.

Đánh giá tính cấu trúc

Để phục vụ cho việc phân rã mô-đun, những tính chất dưới đây cần phải quan tâm tới:

- Kích cỡ: Chúng có quá nhỏ hay quá lớn không? (Các tiêu chuẩn cần phải được thiết lập.) Kích cỡ thích hợp sẽ khác nhau tùy theo ngôn ngữ được sử dụng (khoảng 300 bước với COBOL).
- Chức năng: Có chức năng nào không cần thiết hay không? Có chức năng nào phức hợp hay không? (Nếu có các chức năng phức hợp, cần phải tiếp tục phân rã chúng.)
- Giao diện: Có quá nhiều tham số không? (Trong trường hợp như vậy, cần xem lại quá trình phân rã.)

Đánh giá độ độc lập

Để đánh giá độ độc lập giữa các mô-đun, cho hai độ đo – độ bền của mô-đun và độ móc nối của mô-đun. Việc phân rã được coi là tốt nếu các mô-đun có độ độc lập cao. Độ móc nối mô-đun[46] càng yếu hoặc độ bền của mô-đun[47] càng cao thì độ độc lập giữa các mô-đun càng cao.

Loại độ bền	Độ bền	Độ độc lập	Độ móc nối	Loại móc nối
Độ bền trùng khớp	Yếu	Thấp	Mạnh	Móc nối nội dung
Độ bền logic				Móc nối chung
Độ bền phân loại				Móc nối ngoài
Độ bền thủ tục				Móc nối điều khiển
Độ bền trao đổi				Móc nối nhãn
Độ bền thông tin				Móc nối dữ liệu

[46] **Độ móc nối mô-đun (module coupling)**: Là tiêu chuẩn để đánh giá các mô-đun liên hệ với nhau ở mức độ nào; độ móc nối mô-đun càng yếu thì chúng càng độc lập với nhau.

[47] **Độ bền của mô-đun (module strength)**: Là tiêu chuẩn để đánh giá các thành phần bên trong mô-đun có mối quan hệ với nhau ở mức độ nào; nếu độ bền của mô-đun càng cao thì chúng càng độc lập với nhau.

Độ bền chức năng	Mạnh	Cao	Yếu	

Độ bền của mô-đun có 7 mức được chỉ ra trong bảng dưới đây.

Độ bền của mô-đun	Nội dung
Độ bền trùng khớp	Chương trình được chia ra một cách đơn giản hoặc các chức năng giống hệt nhau sẽ được loại trừ. Không có mối quan hệ đặc biệt giữa các chức năng ở trong mô-đun.
Độ bền logic	Mô-đun có nhiều chức năng liên quan và lựa chọn việc xử lí dựa trên các điều kiện tham số (đối số).
Độ bền kinh điển	Mô-đun hợp nhất nhiều mô-đun được thực thi ở một thời điểm xác định và thực thi nhiều chức năng một cách tuần tự. Mô-đun thiết lập ban đầu là một ví dụ.
Độ bền thủ tục	Mô-đun thực thi nhiều chức năng tuần tự; mối quan hệ bên trong mô-đun là chặt chẽ, và nhiều chức năng không thể được thực thi độc lập với nhau.
Độ bền trao đổi	Mô-đun thực thi các chức năng tuần tự đúng như trong độ bền thủ tục, nhưng dữ liệu được trao đổi giữa các chức năng.
Độ bền thông tin	Mô-đun hợp nhất các chức năng xử lý cùng cấu trúc dữ liệu, có một điểm đầu vào và một điểm đầu ra cho mỗi chức năng, và có thể gọi mỗi chức năng một cách riêng rẽ.
Độ bền chức năng	Mô-đun chỉ gồm một chức năng, và tất cả các lệnh là để thực thi chức năng đó, vì vậy chúng có mối liên hệ rất chặt chẽ.

Độ móc nối mô-đun có 6 mức như dưới đây.[48]

Độ móc nối mô-đun	Nội dung
Móc nối dữ liệu	Chỉ có dữ liệu cần cho việc xử lí mới được truyền vào. Cả các mô-đun đang gọi và mô-đun sẽ được gọi đều không có các mối quan hệ chức năng.
Móc nối nhãn	Chính cấu trúc dữ liệu được truyền vào như các tham số. Mô-đun sắp được gọi sẽ sử dụng một phần của cấu trúc này.
Móc nối điều khiển	Mã chức năng được truyền vào một chương trình con như một tham số, và ảnh hưởng tới việc thực thi của chương trình con.
Móc nối ngoài	Dữ liệu được khai báo bên ngoài sẽ được chia sẻ. Sự khác biệt so với móc nối chung chỉ là trong móc nối ngoài, dữ liệu cần thiết được khai báo bên ngoài.
Móc nối chung	Dữ liệu được định nghĩa trong lĩnh vực chung sẽ được chia sẻ.
Móc nối nội dung	Một mô-đun tham chiếu trực tiếp mô-đun khác và thay đổi nó.

[48] (FAQ) Có nhiều câu hỏi thi về độ độc lập của các mô-đun. Hãy chắc rằng bạn hiểu đúng nội dung về độ bền và độ móc nối mô-đun.

3.2.5 Lập trình

Điểm chính	➢ Cấu trúc hoá logic sẽ cải thiện tính hiệu quả và khả năng bảo trì. ➢ Chương trình chỉ được viết với các cấu trúc điều khiển cơ bản.

Việc lập trình yêu cầu logic có cấu trúc. Với việc cấu trúc hoá logic, mức độ phức tạp có thể được giảm đi và các chương trình có thể được viết một cách dễ hiểu. Điều này giúp nâng cao hiệu quả và cải thiện khả năng bảo trì trong quá trình phát triển.

Định lí cấu trúc

"Trong một chương trình hợp lệ, trong đó có một cặp lối vào và lối ra, không có lặp vô hạn và không có lệnh nào không được thực thi, ta có thể viết logic chương trình bằng cách chỉ sử dụng 3 cấu trúc điều khiển cơ bản: tuần tự, lựa chọn (quyết định), và lặp." Đây được gọi là *Định lí cấu trúc*. Nó rất quan trọng trong việc giữ tính cấu trúc khi viết chương trình. Nguyên lí cơ bản là viết một "chương trình không goto" (chương trình không sử dụng câu lệnh "goto").

Các cấu trúc điều khiển cơ bản

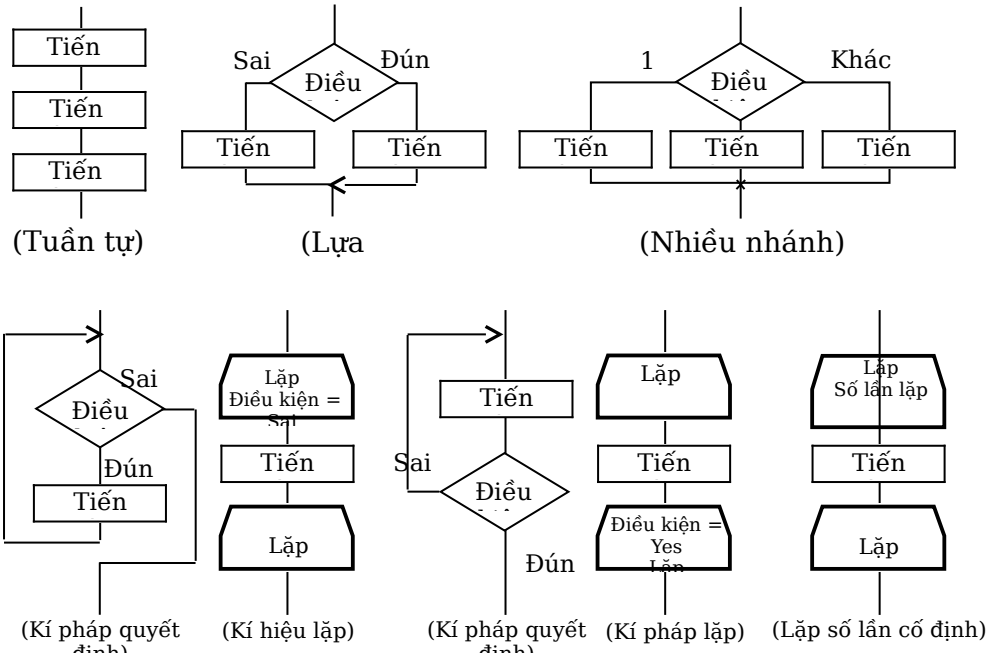
Định lí cấu trúc đưa ra 3 cấu trúc điều khiển cơ bản, nhưng trong thực tế, 6 cấu trúc điều khiển cơ bản dưới đây đều được sử dụng.[49]

Tên	Giải thích
Tuần tự	Chương trình bao gồm các câu lệnh được thực thi tuần tự mà không có câu lệnh "goto" và các quyết định logic.
Lựa chọn (Kiểu "if- then-else")	Hàm được thực hiện phụ thuộc vào việc có theo một điều kiện cụ thể nào hay không.
Nhiều nhánh (Kiểu "case")	Nhiều nhánh được chỉ định phụ thuộc vào giá trị của một biến hay một kết quả xử lí.
Lặp Do-while (Lặp kiểm tra trước)	Điều kiện được quyết định khi bắt đầu quá trình lặp (lặp khi thỏa mãn điều kiện); phụ thuộc vào điều kiện, quá trình lặp có thể không xảy ra lần nào.
Lặp Repeat-until (Lặp kiểm tra sau)	Điều kiện được quyết định khi kết thúc một quá trình lặp (vòng lặp sẽ dừng lại nếu thoả mãn điều kiện); dù là điều kiện nào thì quá trình lặp cũng xảy ra ít nhất một lần.
Lặp số lần cố định	Tiến trình được lặp lại với số lần thực hiện vòng lặp xác định.

Dưới đây là tập các lưu đồ biểu diễn các cấu trúc điều khiển cơ bản ở trên một cách chi tiết.[50]

[49] (Gợi ý) Ba cấu trúc điều khiển cơ bản là "tuần tự", "lựa chọn", và "lặp". "Tuần tự" và "lựa chọn" như đã trình bày trong bảng, còn về "lặp", được đưa ra đầu tiên có nghĩa là lặp do-while.

[50] (Gợi ý) Kí pháp lặp cho biết điều kiện kết thúc vòng lặp. Nói cách khác, nó là điều kiện khi



Các kết quả của thiết kế mô-đun được biểu diễn bằng cách sử dụng các lưu đồ. Tuy nhiên, có nhiều vấn đề liên quan đến các biểu đồ này, và một số công ty sử dụng các phương pháp khác để biểu diễn. Tuy nhiên, trong các kì thi Kĩ sư Công nghệ thông tin, hầu như sử dụng các lưu đồ.

Biểu diễn đồ họa cho thiết kế mô-đun

Bên cạnh các lưu đồ còn có các phương pháp khác biểu diễn kết quả của thiết kế mô-đun như được trình bày trong bảng dưới đây.

Phương pháp	Giải thích
Mã giả (Ngôn ngữ giả)	Mã giả tương tự như một mã chương trình thật, nhưng nó cho phép sử dụng ngôn ngữ tự nhiên (chẳng hạn như tiếng Anh) viết tắt cho các hàm.
Bảng quyết định	Mối quan hệ giữa các điều kiện và nội dung của việc xử lí được biểu diễn dưới dạng bảng.
Biểu đồ NS	Cấu trúc logic được biểu diễn mà không cần sử dụng các mũi tên. Vì hỗ trợ trực quan nên biểu đồ này rất dễ đọc.
Biểu đồ cấu trúc	Một cấu trúc cây được sử dụng để biểu diễn logic.[51]

3.2.6 Các kiểu và thủ tục kiểm thử

Điểm	➢ Các kiểu kiểm thử bao gồm kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử hệ thống.

được thỏa mãn thì vòng lặp sẽ dừng lại. Tuy nhiên, trong một vòng lặp do-while, tiến trình được lặp lại sẽ được thực thi đến khi nào điều kiện còn thỏa mãn, vì vậy ta phải cẩn thận khi biểu diễn điều kiện sử dụng kí hiệu lặp và kí hiệu quyết định.

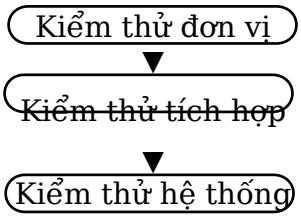
[51] **Biểu đồ cấu trúc**: Biểu đồ này cũng được biết đến như một biểu đồ cấu trúc cây. Để có được lập trình có cấu trúc, những người phát triển và các viện nghiên cứu đã đề xuất ra nhiều loại biểu đồ. Một vài ví dụ nổi tiếng là PAD (Hitachi), YAC (Fujitsu) và HCP (NTT).

chính	➢ Thứ tự của việc kiểm thử có thể từ trên xuống (top-down) hoặc từ dưới lên (bottom-up).

Trong quá trình phát triển hệ thống, công việc thiết kế được thực hiện từ trên xuống, nhưng việc kiểm thử lại được thực hiện từ dưới lên. Nói cách khác, ta dùng cách tiếp cận từ chi tiết đi lên tổng quan. Việc này gọi là tích hợp bậc thang. Trong kiểm thử tích hợp, để thực hiện việc kiểm thử có hiệu quả, ta phải chọn thứ tự kiểm thử các mô-đun một cách cẩn thận.

Thứ tự kiểm thử

Việc kiểm thử được đặt trong thứ tự sau



Kiểm thử một mô-đun ở một thời điểm

Kiểm thử trong đơn vị chương trình (liên kết các mô-đun)

Kiểm thử toàn bộ hệ thống (toàn bộ chương trình)

Kiểm thử đơn vị

Kiểm thử đơn vị là một phép kiểm thử chất lượng của các mô-đun (các đơn vị nhỏ nhất trong chương trình). Khi kiểm thử toàn bộ chương trình, việc tìm ra nguyên nhân gây lỗi là khó khăn, vì vậy kiểm thử đơn vị được thực hiện cho mỗi mô-đun như một đơn vị. Trong kiểm thử đơn vị, ta thực hiện kiểm thử chức năng[52] và kiểm thử cấu trúc.[53] Vì các mô-đun không tự hoạt động nên ta phải chuẩn bị các bộ điều khiển (driver) và các cuống (stub).

← Mô phỏng các chức năng ở các mô-đun mức cao.

← Mô-đun sẽ được kiểm thử

← Mô phỏng các chức năng của các mô-đun mức thấp.

Kiểm thử tích hợp

Với nhiều mô-đun được liên kết với nhau, ta kiểm thử những chương trình đã được liên kết (các mô-đun tải). Mục đích chính ở đây là để kiểm tra giao diện giữa các mô-đun và đầu vào, đầu ra.

Các phương pháp của kiểm thử tích hợp bao gồm kiểm thử từ trên xuống, kiểm thử từ dưới lên, kiểm thử big-bang,[55] và kiểm thử hỗn hợp (sandwich).[56]

Kiểm thử hệ thống

Đây là kiểm thử cuối cùng được thực hiện trong việc phân chia phát triển và được sử dụng để kiểm tra các đặc tả yêu cầu có được thỏa mãn hay không. Ví dụ, ta đặt câu hỏi như "Có vấn đề về việc thực thi không (kiểm thử thực thi)?" và "Nó có thể chịu được các tải nặng không (kiểm thử tải)?" Ta cũng kiểm thử các phần tử ngoại lệ và số đo được đưa ra khi xảy ra lỗi.

---

[52] **Kiểm thử chức năng**: Đây là một loại kiểm thử thẩm định dựa trên các đặc tả của mô-đun để xác nhận rằng tất cả các chức năng được đưa ra cho mô-đun đều thỏa mãn.

[53] **Kiểm thử cấu trúc**: Đây là một loại kiểm thử thẩm định, dựa trên các đặc tả của mô-đun và chương trình nguồn để xác nhận rằng logic của mô-đun là hợp lí.

[54] (Gợi ý) Bộ điều khiển là một chương trình mô phỏng các chức năng của một mô-đun mức cao, và cuống (stub) là một chương trình mô phỏng các chức năng của mô-đun mức thấp. Thông thường, cuống (stub) sẽ trả về một giá trị và do đó nó dễ dàng thực hiện, trong khi đó bộ điều khiển sẽ điều khiển các lời gọi và vì vậy thường phức tạp.

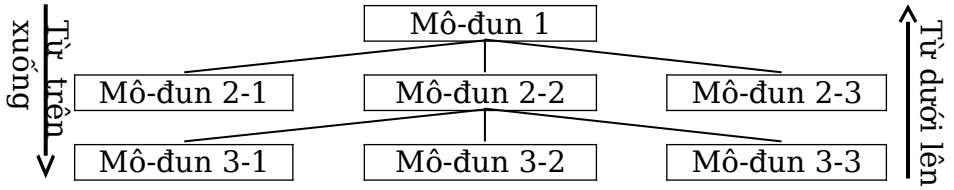
[55] **Kiểm thử big-bang**: Đây là phép kiểm thử trong đó mọi các khía cạnh của các mô-đun đã hoàn thành kiểm thử đơn vị được liên kết lại và kiểm thử. Nếu chương trình ở qui mô nhỏ, nó có thể giảm số thủ tục kiểm thử; tuy nhiên, nếu một lỗi xảy ra, sẽ khó nhận ra được lỗi xảy ra ở đâu.

[56] **Kiểm thử hỗn hợp (sandwich)**: Là phương pháp kiểm thử mà các mô-đun mức thấp được kiểm thử từ dưới lên và các mô-đun mức cao được kiểm thử từ trên xuống. Đây là loại kiểm thử có tính thực tế nhất.

Thứ tự của kiểm thử tích hợp

Kiểm thử tích hợp có thể được thực hiện từ trên xuống hoặc từ dưới lên. Đặc điểm của mỗi kiểu kiểm thử này được minh họa trong bảng và hình vẽ dưới đây.[57]

Kiểu	Đặc điểm
Kiểm thử từ trên xuống	Kiểm thử từ các mô-đun mức cao xuống các mô-đun mức thấp. Cần phải có các cuống (stub) để mô phỏng các mô-đun mức thấp chưa được kiểm thử. Giao diện giữa các mô-đun có thể được kiểm thử một cách đầy đủ. Ban đầu, làm việc song song sẽ khó khăn. Có hiệu quả trong các hệ thống mới được phát triển.
Kiểm thử từ dưới lên	Kiểm thử từ các mô-đun mức thấp lên các mô-đun mức cao. Cần phải có các bộ điều khiển để mô phỏng các mô-đun mức cao chưa được kiểm thử. Các chức năng của chương trình có thể được kiểm thử một cách đầy đủ. Làm việc song song có thể thực hiện ngay từ giai đoạn đầu của việc kiểm thử. Hiệu quả trong việc phát triển các hệ thống mới bằng cách chỉnh sửa các hệ thống có sẵn.



3.2.7 Các kĩ thuật kiểm thử

Điểm chính	➢ Các kĩ thuật kiểm thử bao gồm kiểm thử hộp đen và kiểm thử hộp trắng. ➢ Kiểm thử hộp đen không được sử dụng trong kiểm thử đơn vị.

Mục đích của việc kiểm thử một chương trình là để xác nhận rằng chương trình hoạt động theo các đặc tả và để loại trừ lỗi có trong chương trình. Với mục đích này, một số dữ liệu gây lỗi có thể được cố tình nhập vào. Có 2 kĩ thuật kiểm thử được đề xuất: kiểm thử hộp đen và kiểm thử hộp trắng.

Kiểm thử hộp đen

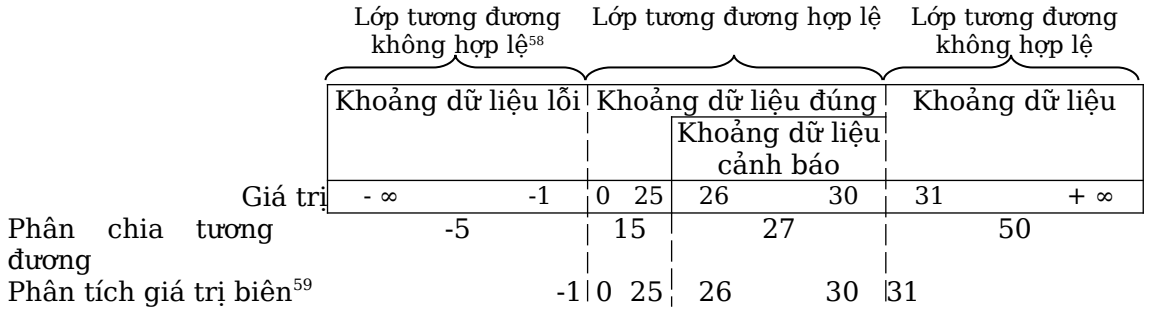
Kiểm thử hộp đen là phương pháp mà nhờ đó các trường hợp kiểm thử được thiết kế dựa trên các đặc tả bên ngoài của chương trình. Không quan tâm đến logic của chương trình, dữ liệu kiểm thử được chuẩn bị chỉ dựa trên các đặc

[57] (FAQ) Thường có những câu hỏi như "Đâu là mô tả thích hợp cho kiểm thử từ trên xuống?" và "Đâu là mô tả thích hợp cho kiểm thử từ dưới lên?" Hãy chắc rằng bạn hiểu sự khác nhau giữa các phương pháp kiểm thử từ trên xuống và từ dưới lên cũng như vai trò của cuống (stub) và bộ điều khiển (driver).

tả bên ngoài. Các tiêu chí kiểm thử được đưa tra trong bảng dưới đây.

Tên	Giải thích
Phân chia tương đương	Khoảng của các giá trị đầu vào được phân chia thành nhiều lớp, và một giá trị kiểm thử sẽ được chọn ra từ mỗi lớp như một giá trị đại diện (ví dụ, giá trị trung bình của lớp).
Phân tích giá trị biên	Khoảng của các giá trị đầu vào được phân chia thành nhiều lớp, và các giá trị biên (giá trị giới hạn) của mỗi lớp được chọn ra và trở thành giá trị kiểm thử.

Ví dụ, giả sử rằng trong một mục số (giá trị nguyên), từ 0 đến 30 là các giá trị dữ liệu hợp lệ còn các số nguyên khác là sai và đưa ra một thông báo lỗi. Hơn nữa, giả sử rằng các giá trị từ 26 đến 30 đưa ra lời cảnh báo sẽ được hiển thị như những giá trị cảnh báo. Chẳng hạn như ở đây, dưới tiêu chí phân chia tương đương, có thể chọn ra tập dữ liệu kiểm thử là (-5, 15, 27, 50). Dưới tiêu chí phân tích giá trị biên, có thể chọn ra tập các giá trị biên của các lớp là (-1, 0, 25, 26, 30, 31).



Kiểm thử hộp trắng

Kiểm thử hộp trắng là phương pháp mà nhờ đó các trường hợp kiểm thử được thiết kế dựa trên các đặc tả bên trong của chương trình. Dưới đây là một số tiêu chí kiểm thử: bao phủ lệnh, bao phủ điều kiện quyết định (bao phủ nhánh), bao phủ điều kiện (bao phủ điều kiện nhánh), bao phủ quyết định/điều kiện, và bao phủ đa điều kiện. Các tiêu chí này được liệt kê theo chiều tăng của tính khắt khe (tính nghiêm ngặt).[60]

Tên	Giải thích
Bao phủ lệnh	Các trường hợp kiểm thử được thiết kế sao cho mỗi câu lệnh được thực hiện ít nhất một lần.
Bao phủ điều kiện quyết định (bao phủ nhánh)	Các trường hợp kiểm thử được thiết kế sao cho các trường hợp đúng và sai trong quyết định đều được thực hiện ít nhất một lần.
Bao phủ điều kiện (bao phủ điều kiện nhánh)	Các trường hợp kiểm thử được thiết kế sao cho khi có nhiều điều kiện, mọi tổ hợp cho các trường hợp đúng và sai được thỏa mãn.
Bao phủ quyết định/điều kiện	Là kết hợp của bao phủ nhánh và bao phủ điều kiện.

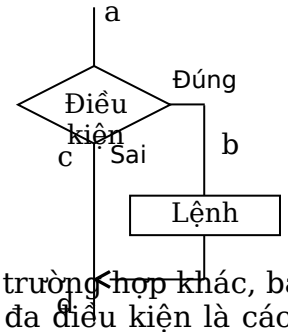
[58] **Lớp tương đương hợp lệ/ Lớp tương đương không hợp lệ**: Trong kiểm thử hộp đen, khoảng của các giá trị dữ liệu đúng được gọi là lớp tương đương hợp lệ, và khoảng của các giá trị dữ liệu sai được gọi là lớp tương đương không hợp lệ.

[59] (FAQ) Có các câu hỏi mà ở đó bạn cần chuẩn bị dữ liệu kiểm thử cho tiêu chí phân chia tương đương và phân tích giá trị biên. Hãy hiểu một cách đầy đủ ý nghĩa của các thuật ngữ đó, và chắc rằng bạn có thể chuẩn bị được dữ liệu kiểm thử.

[60] (FAQ) Tất cả các kì thi đều có các câu hỏi về ý nghĩa của kiểm thử hộp đen vào kiểm thử hộp trắng. Hãy chắc rằng bạn biết những ý nghĩa đó.

Bao phủ đa điều kiện	Các trường hợp kiểm thử được thiết kế sao cho mọi tổ hợp của các trường hợp đúng/sai trong mọi điều kiện đều được kiểm thử.

Ví dụ, giả sử có một chương trình có cấu trúc như hình dưới đây. Dữ liệu được chuẩn bị cho mỗi tiêu chí kiểm thử như sau:
Trong bao phủ lệnh, dữ liệu đi qua đường "a, b, d" được chuẩn bị vì đường này đi qua tất cả các lệnh. Nói cách khác, chỉ có dữ liệu theo trường hợp "Đúng" của "điều kiện" được chuẩn bị. Trong bao phủ điều kiện quyết định, dữ liệu cho các trường hợp "Đúng" và "Sai", tức là dữ liệu qua cả hai đường "a, b, d" và "a, c, d", đều được chuẩn bị.



Các trường hợp khác, bao phủ điều kiện, bao phủ quyết định/điều kiện và bao phủ đa điều kiện là các kĩ thuật được sử dụng cho trường hợp có nhiều điều kiện. Ví dụ, giả sử rằng ta xem xét đến đa điều kiện "a và b" ở đây.

Số	a	b	a và b	Bao phủ điều kiện	Bao phủ quyết định/điều kiện	Bao phủ đa điều kiện
(1)	Đúng	Đúng	Đúng		X	X
(2)	Đúng	Sai	Sai	X	X	X
(3)	Sai	Đúng	Sai	X	X	X
(4)	Sai	Sai	Sai			X

Trong bao phủ điều kiện,[61] các số (2) và (3) sẽ được kiểm thử như là các tổ hợp cho các trường hợp đúng và sai trong đa điều kiện. Tuy nhiên, đa điều kiện "a và b" là sai trong cả hai số trên nên trường hợp cả hai cùng đúng sẽ không cần kiểm thử. Trong bao phủ quyết định/điều kiện, sẽ có trường hợp cả hai cùng đúng, do vậy cả (1), (2) và (3) đều được kiểm thử. Trong bao phủ đa điều kiện, mọi tổ hợp (1), (2), (3) và (4) đều được kiểm thử.

[61] (FAQ) Liên quan đến bao phủ lệnh vào bao phủ điều kiện quyết định, các câu hỏi thường hỏi dữ liệu cụ thể. Tốt nhất là hãy thực sự chuẩn bị dữ liệu kiểm thử ra kiểm tra.

Câu hỏi nhanh

Q1 Liệt kê các kiểu công việc được thực hiện trong thiết kế ngoài.

Q2 Liệt kê các kiểu công việc được thực hiện trong thiết kế trong.

Q3 Để tăng mức độc lập của các mô-đun, cần phải làm gì với độ bền của mô-đun và độ móc nối mô-đun? Kể tên loại độ bền và độ móc nối được nói đến ở đây.

Q4 Mô tả ngắn gọn mỗi phương pháp kiểm tra sau: "kiểm tra số", "kiểm tra định dạng", "kiểm tra giới hạn", "kiểm tra khoảng", và "kiểm tra tuần tự".

Q5 Mô tả các đặc điểm của kiểm thử hộp đen.

Q6 Mô tả các đặc điểm của kiểm thử hộp trắng.

A1

- Kiểm chứng phân tích yêu cầu (Kiểm tra các yêu cầu của người dùng)
- Xác định và phát triển các hệ thống con (Chia hệ thống thành các đơn vị chức năng)
- Thiết kế màn hình, thiết kế biểu mẫu (Thiết kế định dạng đầu vào, đầu ra)
- Thiết kế mã (Các cấu trúc mã nhân viên, mã sản phẩm,…)
- Thiết kế dữ liệu logic (Xác định dữ liệu; quyết định cấu trúc dữ liệu và các mục)
- Chuẩn bị các đặc tả thiết kế ngoài

A2

- Phân rã, cấu trúc hóa chức năng (Xác định các chức năng và gom nhóm chúng dựa trên các nội dung xử lí)
- Thiết kế tệp (Quyết định cụ thể phương tiện, tổ chức, bố cục,… của tệp)
- Thiết kế chi tiết vào/ra (Quyết định phương tiện, phương thức và phương pháp kiểm tra đầu vào/đầu ra)
- Chuẩn bị các đặc tả thiết kế trong

A3

Cần tăng độ bền của mô-đun (độ bền chức năng) và giảm độ móc nối mô-đun (móc nối dữ liệu)

A4

Kiểm tra số: Kiểm tra phần tử số có thực sự là một số hay không
Kiểm tra định dạng: Kiểm tra dữ liệu có đúng định dạng không và các con số có đúng thứ tự hay không
Kiểm tra giới hạn: Kiểm tra giá trị có nằm trong giới hạn trên và giới hạn dưới không
Kiểm tra khoảng: Kiểm tra giá trị có nằm trong đúng khoảng hay không
Kiểm tra tuần tự: Kiểm tra các phần tử khóa có được liệt kê theo thứ tự hay không

A5

Đây là phương pháp mà nhờ đó các trường hợp kiểm thử được thiết kế dựa trên các đặc tả bên ngoài của chương trình. Không quan tâm đến logic của chương trình, dữ liệu kiểm thử được chuẩn bị dựa trên các đặc tả bên ngoài.

A6

Đây là phương pháp mà nhờ đó các trường hợp kiểm thử được thiết kế dựa trên các đặc tả bên trong chương trình. Có một số tiêu chí kiểm thử, xếp theo chiều tăng của tính khắt khe (tính nghiêm ngặt): bao phủ lệnh, bao phủ điều kiện quyết định (bao phủ nhánh), bao phủ điều kiện (bao phủ điều kiện nhánh), bao phủ quyết định/điều kiện, và bao phủ đa điều kiện.

Câu hỏi 1

Độ khó: ** Tần suất: **

Q1. Phát biểu nào dưới đây là thích hợp khi quan tâm tới việc tối ưu hóa một trình biên dịch?

a) Nó tạo ra mã trung gian cho trình thông dịch thay vì tạo ra mã đích.
b) Nó tạo ra mã đích chạy trên một bộ máy khác, không phải máy tính mà ở đó trình biên dịch chạy.
c) Nó tạo ra mã đích hiển thị tên của chương trình con mà việc điều khiển được đưa tới hoặc nội dung của một biến ở thời điểm cụ thể khi chương trình được thực thi.
d) Nó phân tích chương trình nguồn và tạo ra mã đích, nhờ vậy việc xử lí có thể hiệu quả hơn trong quá trình thực thi.

Đáp án câu 1

Đáp án đúng: d

Việc tối ưu hóa một trình biên dịch có nghĩa là loại bỏ những phần dư thừa của chương trình đích.

a) Tối ưu hóa là loại bỏ các phần dư thừa; trình biên dịch tạo ra mã trung gian. Vì vậy, phát biểu này không phải là một lời giải thích cho sự tối ưu hóa.

b) Đây là lời giải thích cho các trình biên dịch chéo.

c) Đây là lời giải thích cho các bộ dò vết.

d) Việc tối ưu hóa làm tăng hiệu quả xử lí trong quá trình thực thi, có nghĩa là nó bao gồm việc loại bỏ các dấu ngoặc đơn không cần thiết và các thao tác trước tính toán chỉ liên quan đến các hằng số.

Câu hỏi 2

Độ khó: ** Tần suất: ***

Q2. Phát biểu nào dưới đây là thích hợp cho phương pháp định nghĩa một phần tử - đơn vị nhỏ nhất trong việc tạo nên một văn bản XML?

a) Một thẻ mở và một thẻ đóng luôn đi đôi với nhau trong cấu trúc, không được thiếu bất kì thẻ nào.

b) Một phần tử dữ liệu được tạo ra sao cho nó có thể được đặt giữa một thẻ mở và một thẻ đóng. Tuy nhiên, trong một số trường hợp, không tồn tại dữ liệu.

c) Trong một tài liệu XML, nhiều phần tử gốc có thể được định nghĩa để biểu diễn cấu trúc phân cấp.

d) Thông tin chú thích được thêm vào để biểu diễn kiểu của phần tử. Nó được định nghĩa như tên của phần tử.

Đáp án câu 2

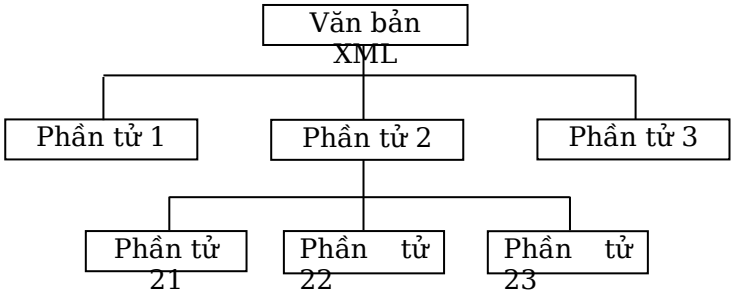
Đáp án đúng: b

XML (eXtensible Markup Language – Ngôn ngữ đánh dấu mở rộng) mở rộng các chức năng của HTML, loại bỏ các chức năng không cần thiết của SGML và được tối ưu hóa để sử dụng trên Web. Giống như HTML, XML cũng được sử dụng trên Web; điểm khác biệt là trong khi các thẻ của HTML là cố định thì các thẻ của XML có thể định nghĩa bởi người dùng thông qua việc định nghĩa DTD (Document Type Definition – Định nghĩa kiểu tài liệu).

a) Đúng là cấu trúc có dữ liệu được bao quanh bởi thẻ mở và thẻ đóng, nhưng khi không có dữ liệu, để biểu diễn một phần tử trống, một thẻ đặc biệt giống như <tên phần tử/> có thể được sử dụng và được phân biệt với thẻ mở. Ví dụ, có thể mô tả giống như . Do vậy, thẻ mở và thẻ đóng không nhất thiết phải đi đôi với nhau.

b) Theo nguyên tắc, dữ liệu được bao quanh bởi một thẻ mở và một thẻ đóng. Nếu không có dữ liệu, điều này cũng có thể chấp nhận được.

c) Trong XML, tất cả các phần tử được đặt trong cấu trúc lồng nhau. Một phần tử có thể chứa nhiều phần tử khác, nhưng không có phần tử nào được chứa trong nhiều phần tử. Ở đây, phần tử chứa các phần tử khác được gọi là “cha,” và phần tử được chứa bên trong gọi là “con.”



d) Thông tin chú thích không được chứa trong dữ liệu và trong thực tế có thể bỏ qua. Các chú thích được bao quanh bởi các thẻ “<!--” và “-->.”

Câu hỏi 3

Độ khó: * Tần suất: ***

Q3. Phát biểu nào sau đây mô tả tính chất của mô hình thác đổ đảm bảo tính nhất quán của phát triển hệ thống?

a) Theo nguyên tắc, không cho phép đi ngược lại qua các pha phát triển.
b) Phát triển hệ thống được chia thành nhiều pha để quản lí.
c) Nhất định cần tạo ra một tổ chức dự án.
d) Các hoạt động phát triển trong pha tiếp theo dựa trên kết quả được đưa xuống từ pha trước đó.

Đáp án câu 3

Đáp án đúng: d

Mô hình thác đổ là một mô hình qui trình trong đó việc phát triển hệ thống thực hiện từ pha trên (thượng lưu) xuống pha dưới (hạ lưu) theo thứ tự: “lập kế hoạch cơ bản → thiết kế ngoài → thiết kế trong → thiết kế chương trình → lập trình → kiểm thử → cài đặt, vận hành, bảo trì”. Vì luồng qui trình phát triển được chia thành các pha nên dễ dàng nắm được tổng quan của toàn bộ dự án. Việc quản lí dự án cũng được coi là dễ dàng hơn vì công việc đi từ trên xuống dưới một cách tuần tự. Tuy nhiên, vì không được quay ngược lại, nó có nhược điểm là hiệu quả của việc phát triển sẽ giảm xuống nếu qui trình đòi hỏi sự hồi qui.

Trong mô hình thác đổ, việc kiểm điểm được thực hiện ở cuối mỗi pha để các lỗi không được mang sang pha tiếp theo. Nếu có một lỗi được tìm ra trong một pha hạ lưu (pha sau pha lập trình), chi phí cho việc điều chỉnh hệ thống (chi phí cho việc hồi qui) là rất lớn. Vì vậy, các lỗi phải được tìm ra ở một pha thượng lưu (một pha thiết kế giữa lập kế hoạch cơ bản và thiết kế chương trình).

a) Đây là một trong những đặc điểm của mô hình thác đổ, nhưng chỉ đơn giản là mô tả tiến trình thực hiện ra sao; nó không đảm bảo sự nhất quán của phát triển hệ thống.

b) Đây là một trong những đặc điểm của mô hình thác đổ, nhưng nó mô tả việc phân loại nội dung của các hoạt động; nó không đảm bảo sự nhất quán của phát triển hệ thống.

c) Một đội dự án được tổ chức cho việc phát triển hệ thống nói chung, không chỉ trong mô hình thác đổ.

d) Pha thiết kế của mô hình thác đổ là sự cải tiến bậc thang. Nội dung của pha trước được đưa vào pha tiếp theo; điều này đảm bảo sự nhất quán của phát triển hệ thống.

Câu hỏi 4

Độ khó: * Tần suất: ***

Q4. Phát biểu nào dưới đây thích hợp với mô hình chế thử, một phương pháp phát triển phần mềm?

a) Vì các hoạt độc thực hiện tuần tự từ lập kế hoạch cơ bản, thiết kế ngoài, thiết kế trong, thiết kế chương trình, lập trình, và kiểm thử nên có thể có được cái nhìn tổng quan về toàn bộ dự án và dễ dàng quyết định lịch trình cũng như cấp phát tài nguyên.
b) Vì một mô hình thử nghiệm được tạo ra ở pha sớm của phát triển hệ thống nên nó có thể loại bỏ sự không chắc chắn và sự khác biệt trong suy nghĩ của bộ phận khách hàng và bộ phận phát triển.
c) Các đặc điểm của phần mềm được chia thành nhóm mà các đặc tả là cố định, không yêu cầu thay đổi và nhóm mà các đặc tả yêu cầu thay đổi. Sau đó, qui trình cho việc tạo ra, kiểm điểm và thay đổi mã theo những đặc tả đó sẽ được lặp lại.
d) Một ứng dụng lớn được chia thành các thành phần có tính độc lập cao; sau đó các qui trình của việc thiết kế, lập trình và kiểm thử được lặp lại cho các thành phần đó, và dần dần mở rộng phạm vi của việc phát triển chương trình.

Đáp án câu 4

Đáp án đúng: b

Chế thử là một phương pháp mà ở đó bản mẫu (mô hình thử nghiệm) được làm thành các phần có thể thấy rõ ràng từ người dùng hệ thống (màn hình, biểu mẫu,...) và các hệ thống được phát triển dựa trên phản hồi của người dùng – những người đã kiểm tra bản mẫu. Do đó, phát biểu b) là thích hợp.
a) giải thích mô hình thác đổ, và d) giải thích mô hình xoắn ốc.

Câu hỏi 5

Độ khó: * Tần suất: ***

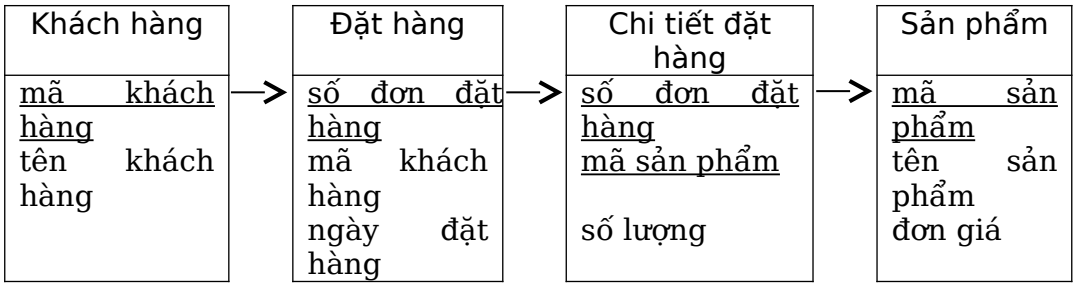
Q5. Mối quan hệ nào dưới đây được đưa ra bởi biểu đồ E-R?

a) Mối quan hệ giữa các thực thể
b) Mối quan hệ giữa các kiểu thực thể và các thể hiện của chúng
c) Mối quan hệ giữa dữ liệu và các tiến trình
d) Mối quan hệ giữa các tiến trình

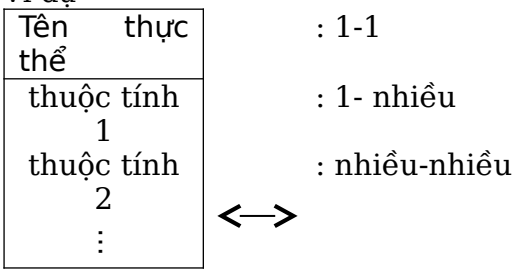
Đáp án câu 5

Đáp án đúng: a

Biểu đồ E-R chỉ ra mối quan hệ giữa các thực thể (các đối tượng thực sự), biểu thị một thực thể với một hình chữ nhật ( ) và các quan hệ tương ứng giữa các thực thể với một đường nối (→, ←→ , —). Dưới đây là một ví dụ của biểu đồ E-R.



Ví dụ



Thuộc tính được gạch chân là một thuộc tính khóa chính. Kiểu thực thể là một thực thể có dữ liệu để quản lí. Thông thường, các thực thể được biểu diễn bằng các danh từ như “khách hàng” và “sản phẩm.” “Các thể hiện” (“Instances”) là các thực thể mang giá trị.

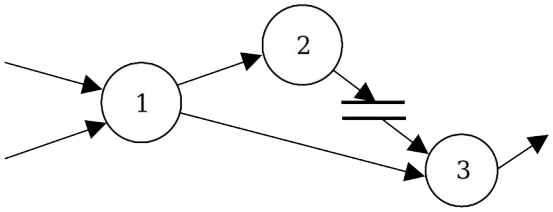
Khách hàng

Mã khách hàng	Tên khách hàng	Địa chỉ khách hàng	
			← Kiểu thực thể
1011	George Bush	Crawford, Texas	← Thể hiện
1021	William Clinton	Hope, Arkansas	← Thể hiện

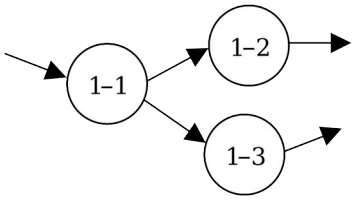
Câu hỏi 6

Độ khó: ** Tần suất: ***

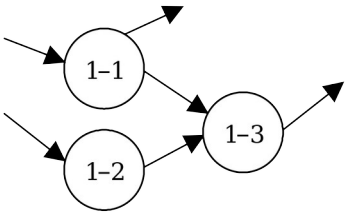
Q6. Hình dưới đây chỉ ra một mức cụ thể của một biểu đồ luồng dữ liệu (DFD) phân cấp. Đâu là phương pháp phù hợp nhất để mô tả DFD của mức ngay bên dưới? Giả sử rằng các tiến trình trong mức ngay bên dưới *Tiến trình n* là các tiến trình được đánh theo dạng *n-1*, *n-2*, *...*



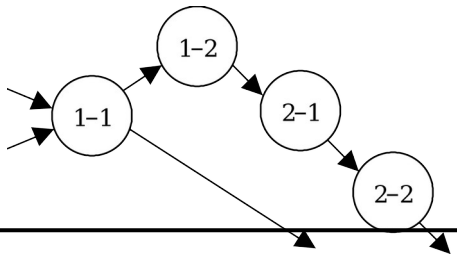
a)



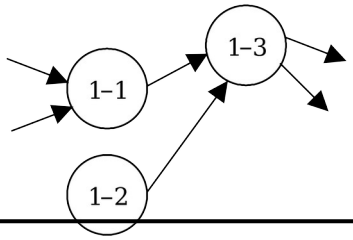
b)



c)



d)



Đáp án câu 6

Đáp án đúng: **b**

Trong biểu đồ luồng dữ liệu, các tiến trình (các vòng tròn) được phân rã ra theo thứ tự. Vì vậy, khi chúng được phân rã, nhiều tiến trình ở mức cao hơn không thể hợp nhất với các tiến trình ở mức thấp hơn. Trong biểu đồ luồng dữ liệu được đưa ra trong câu hỏi này, chú ý rằng *Tiến trình 1* có 2 luồng dữ liệu vào và 2 luồng dữ liệu ra.

a) Trong khi *Tiến trình 1* có 2 luồng dữ liệu vào, mối tiến trình con của nó lại chỉ có một luồng dữ liệu vào.

b) *Tiến trình con 1-1* có một luồng dữ liệu vào và *Tiến trình con 1-2* cũng vậy. Tổng cộng là 2. Về luồng dữ liệu ra, có một luồng từ *Tiến trình con 1-1* và một luồng từ *Tiến trình con 1-3*, tổng cộng là 2. Do đó, đây có thể là một biểu đồ luồng dữ liệu đã được phân rã trong câu hỏi này.

c) Vì biểu đồ luồng dữ liệu phân rã một tiến trình, các tổ hợp như {(1-1), (1-2)} và {(2-1), (2-2)} được chấp nhận, trong khi đó tổ hợp như {(1-1), (1-2), (2-1), (2-2)} thì không, ở đó nhiều tiến trình ở mức cao hơn cũng được kết hợp vào.

d) Số lượng luồng dữ liệu vào và luồng dữ liệu ra là đúng, nhưng mọi tiến trình phải có ít nhất một luồng dữ liệu vào và ít nhất một luồng dữ liệu ra. Ở đây không có luồng dữ liệu vào cho *Tiến trình 1-2*.

Câu hỏi 7

Độ khó: ** Tần suất: ***

Q7. Bảng dưới đây đưa ra số lượng các phần tử theo nhóm và trọng số cho các chức năng người dùng của một chương trình ứng dụng. Thông tin dựa trên phương pháp điểm chức năng. Chương trình ứng dụng này có bao nhiêu điểm chức năng? Ở đây, hệ số hiệu chỉnh của độ phức tạp là 0.75.

Kiểu chức năng người dùng	Số lượng phần tử	Trọng số
Đầu vào từ bên ngoài	1	4
Đầu ra bên ngoài	2	5
Tệp logic bên trong	1	10
Tệp giao diện bên ngoài	0	7
Yêu cầu từ bên ngoài	0	4

a) 18 b) 24 c) 30 d) 32

Đáp án câu 7

Đáp án đúng: a

Trong phương pháp điểm chức năng, số điểm chức năng thu được như sau:

- Nhân số lượng chức năng (số phần tử) với trọng số tương ứng
- Tìm tổng của các kết quả trên
- Nhân tổng với độ phức tạp (hệ số hiệu chỉnh của độ phức tạp) để có được câu trả lời

Do đó số điểm chức năng được tính như dưới đây:
Số điểm chức năng = $(1 * 4 + 2 * 5 + 1 * 10 + 0 * 7 + 0 * 4) * 0.75$
= 18

Câu hỏi 8

Độ khó: *** Tần suất: ***

Q8. Thủ tục nào dưới đây được ưu tiên để cải thiện độ tin cậy và khả năng bảo trì trong thiết kế mô-đun phần mềm?

a) Tăng cả độ bền và độ móc nối mô-đun
b) Tăng độ bền mô-đun trong khi giảm độ móc nối
c) Giảm độ bền mô-đun trong khi tăng độ móc nối
d) Giảm cả độ bền và độ móc nối mô-đun

Đáp án câu 8

Đáp án đúng: b

Trong thiết kế mô-đun, việc tăng độ độc lập mô-đun nên được xem xét đến để cải thiện độ tin cậy và khả năng bảo trì. Nếu các mô-đun độc lập cao, chúng sẽ không bị ảnh hưởng bởi các mô-đun khác, nhờ đó nâng cao độ tin cậy. Hơn nữa, khả năng bảo trì cũng có thể được cải thiện vì việc chỉnh sửa ở một mô-đun sẽ không ảnh hưởng đến các mô-đun khác.

Các tiêu chí đánh giá để đo lường độ độc lập của các mô-đun bao gồm độ bền mô-đun và độ móc nối mô-đun.

Độ bền mô-đun đo mức độ liên kết bên trong mỗi mô-đun. Mối liên kết bên trong mô-đun càng mạnh thì mô-đun càng có độ độc lập cao. Độ móc nối mô-đun đo mức độ liên kết giữa các mô-đun. Mối liên kết giữa các mô-đun càng yếu thì độ độc lập mô-đun càng cao.

Câu hỏi 9

Độ khó: * Tần suất: ***

Q9. Phát biểu nào dưới đây là phù hợp với kiểm thử hộp trắng?

a) Việc kiểm thử được thực hiện tuần tự kết hợp các mô-đun từ mức thấp lên mức cao.
b) Việc kiểm thử được thực hiện tuần tự kết hợp các mô-đun từ mức cao xuống mức thấp.
c) Việc kiểm thử được thực hiện có chú ý tới cấu trúc bên trong của mô-đun.
d) Việc kiểm thử được thực hiện để kiểm tra các chức năng có làm việc theo các đặc tả hay không, và không quan tâm tới cấu trúc bên trong của mô-đun.

Đáp án câu 9

Đáp án đúng: **c**

Kiểm thử hộp trắng là phương pháp tập trung vào luồng điều khiển chương trình, chuẩn bị dữ liệu kiểm thử đi qua các con đường chủ chốt trong chương trình, và thực hiện việc kiểm thử. Vì cấu trúc bên trong và logic của chương trình được kiểm tra kĩ lưỡng, chúng ta có thể kiểm thử các chức năng chi tiết theo quan điểm của lập trình viên, tuy nhiên các chức năng có trong đặc tả nhưng chưa được thực hiện trong chương trình thì không được chọn là dữ liệu thử nghiệm.

a) Đây là giải thích về kiểm thử từ dưới lên.

b) Đây là giải thích về kiểm thử từ trên xuống.

d) Đây là giải thích về kiểm thử hộp đen.